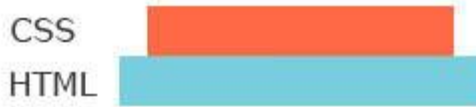


Урок 1: Какво представлява CSS?

CSS идва от съкращението *Cascading Style Sheets* и представлява отделен език, който ни помага да променяме външния вид на нашата xHTML страница. Възуално връзката между HTML и CSS може да бъде представена така:



От графиката виждаме, че CSS е нещо като “надбавка” към HTML т.е. уеб страницата може да съществува сама по себе си дори и без помощта на CSS. CSS обаче не може да съществува без наличието на уеб страница, защото CSS се използва, за да промени стила на желаните HTML елементи.

Забележка Искам да отбележа, че CSS ни позволява да стилизираме елементите, които се намират в `<body>`. Елементите, които се намират в `<head>` не могат да бъдат стилизирани, защото CSS няма достъп до тях, а и те са невидими за потребителя.

Начините на задаване на CSS са три:

- Чрез атрибута *style* в отварящия таг на даден елемент;
- Чрез елемента `<style>` в раздела `<head>` на html документа;
- Чрез външен Style sheet.

Урок 2: Част I

Начини на задаване на CSS

Чрез атрибута *style* в отварящия таг на даден елемент

Този вид стил се нарича инлайн (inline), защото се прилага върху конкретен елемент чрез използването на *style* атрибута в отварящия таг.

Това е първият и може би най-лесният метод за прилагане на CSS към даден елемент. Лесен е, защото задаваме стила в самия елемент. Така не се налага да помним нито вида на елемента, нито структурата на html документа. Всичко което трябва да направим е да използваме атрибута *style*.

Забележка Атрибута *style* е **незадължителен**. Използването му е по избор. Това което *style* атрибута всъщност прави е да казва на брауъра “Третирай всичко между кавичките като CSS стил и го приложи на избрания елемент”.

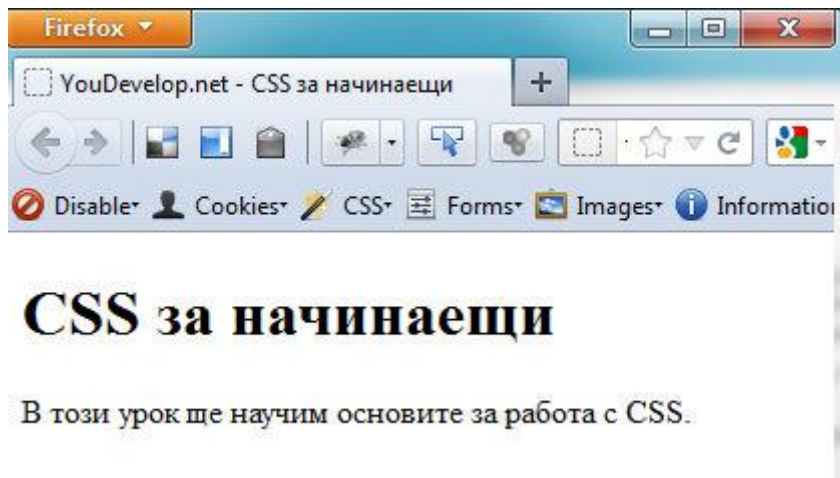
Пример:

Стъпка 1: Нека създадем нова уеб страница като използваме *Strict Doctype*. Също така нека добавим едно h1 заглавие и един текстов параграф. Кода би изглеждал ето така:

```
1. <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
2. "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
3. <html xmlns="http://www.w3.org/1999/xhtml">
4.     <head>
5.         <meta http-equiv="Content-Type" content="text/html;
6.             charset=utf-8" />
7.         <title>YouDevelop.net - CSS за начинаещи</title>
8.     </head>
9.     <body>
10.         <h1>CSS за начинаещи</h1>
11.         <p>В този урок ще научим основите
```

12. за работа с CSS.</p>
13. </body>
14. </html>

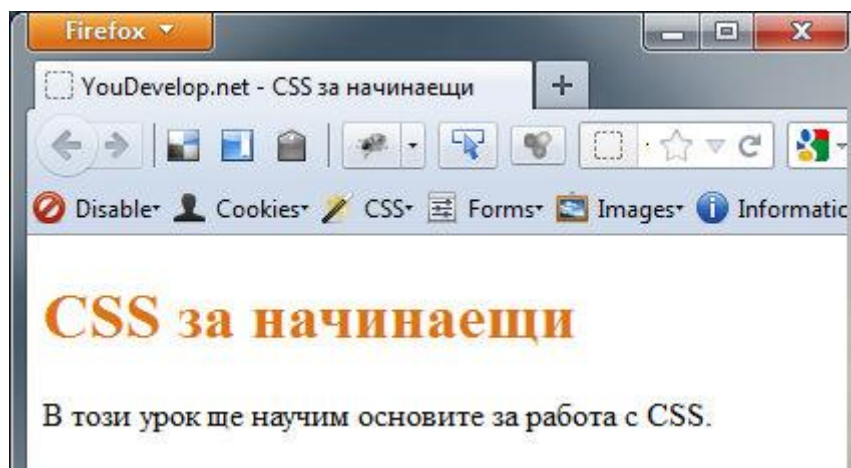
След като запазим документа и го отворим в браузър, страницата ще изглежда по следния начин:



Стъпка 2: По подразбиране цвета на текста на h1 заглавието и на параграфа е черен. Нека го променим. Заглавието се отличава значително от останалата част на страницата, но ние искаме цвета на текста да бъде оранжев. За да постигнем този ефект е необходимо да използваме атрибута *style* в отварящия h1 таг. След това определяме свойството, което искаме да променим (в случая цвета), а като стойност на това свойство въвеждаме избраният цвят. Промени кода на отварящия h1 таг по следния начин:

1. <h1 style="color:#dd7319;">CSS за начинаещи</h1>

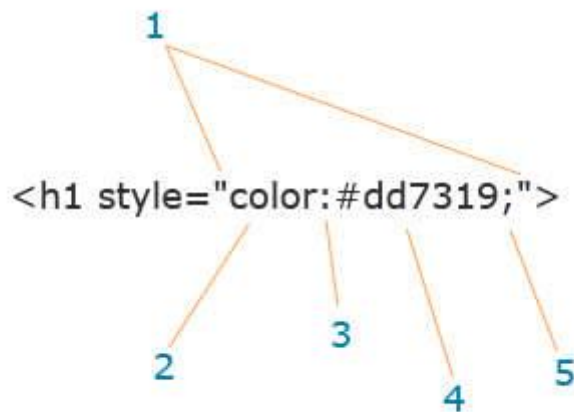
Стъпка 3: Запази промените и отвори страницата с браузър. Резултата трябва да е следния:



По същия начин можеш да промениш цвета на текстовия параграф.

Но какво всъщност стана? Как работи всичко това? Има няколко важни неща, които трябва да запомниш! Нека видим кои са те.

Структура на атрибута style



Всяка част от структурата изпълнява определено значение. Следва описание на всеки един от тези елементи:

1. Кавичките определят границите на *style* атрибута. Всичко, което е между тези кавички се третира от браузъра като CSS и се прилага върху дадения елемент;
2. Общото наименование на двойката **свойство:стойност** се нарича декларация. Първата част от структурата на декларацията се нарича **свойство** (на английски *property*). Свойството е стила, който искаме да променим. Цвят, шрифт, разстояние между редовете и буквите, разстояние между елементите, ширина и височина, всичко това (и много други) са свойства на даден елемент. В този случай свойството, което съм използвал е **color**;
3. Знака **:** определя края на свойството. Това което следва е стойността на избраното свойство;
4. Втората част от декларацията се нарича **стойност** (на английски *value*). Самата дума ни подсказва, че това е стойността на свойството. Тук възможностите са различни като всяко свойство **има точно определени стойности**, които може да приема. Стойности, които не се поддържат от свойството се игнорират. В нашия случай за стойност на **color** съм използвал специален цвят, който е образуван от шестнадесетичен цветови код (на английски *hexadecimal color*). Повече за тези цветове ще научим в някой от бъдещите уроци. Като стойност на **color** можем да използваме и някои от често срещаните английски думи за цвят като *red*, *green*, *orange*, *blue* и т.н. Ако имаш желание пробвай някои от тях.
5. Знака **;** определя края на декларацията. Атрибута *style* ни позволява да използваме няколко декларации наведнъж, като всяка следваща трябва да бъде разделена от предходната, посредством този знак. Например, ако искаме освен цвета да променим и размера на заглавието *h1*, кода ще придобие следния вид:

1. `<h1 style="color:#dd7319; font-size:20px;">`
2. CSS за начинаещи</h1>

По този начин можем да “навържем” колкото искаме декларации, въпреки че прекалено многото в един момент ще ни затруднят при намирането на евентуални грешки в кода.

Обобщение

Разгледахме първият от трите начина за прилагане на CSS в нашата уеб страница. Подробно видяхме и структурата на CSS декларациите. Сега остава да изброя кои са (според мен) предимствата и недостатъците на този метод. Ето ги и тях:

Предимства

- Добавянето на стил върху текущия елемент е изключително лесно. Всичко, което трябва да направиш е да използваш допълнителния атрибут *style*;
- Атрибута *style* може да се използва във всеки един от елементите намиращи се в `<body>` на нашата уеб страница (дори и на самия `<body>` таг);
- **Не е необходимо** да се създава отделен файл (както при използването на външен CSS документ). Стила се въвежда директно в самия елемент.
- Този метод е с **най-висок приоритет**, когато трябва да се определи кой стил да се приложи върху даден елемент. Преоритизирането при прилагане на стилове в CSS е от голямо значение, затова тази тема ще я разгледаме в отделен урок.

Недостатъци

- Ако искаш да приложиш един и същи стил на няколко елемента, трябва да добавиш *style* атрибута на всеки от тях. Например имаш 5 заглавия от h2 ниво и искаш всички те да изглеждат еднакво. В такъв случай трябва да приложиш един и същи стил на всяко заглавие по отделно;
- Много повече работа при поддръжката на кода. Ако разгледаме отново горния пример, ще забележим, че за да променим цвета на заглавието от черен на син, трябва да редактираме кода на 5 различни места. Представи си, че заглавията са повече. Тогава работата се увеличава значително;
- Този вид стил **не може** да се прилага върху няколко html документа наведнъж.
- Използването на прекалено много декларации увеличава възможността за допускане на грешка. Също така кода става труден за четене.
- Миксиране на структурата на страницата със стила ѝ. Както знаем от уроците по HTML, XHTML елементите се използват, за да оформят структурата на уеб страницата. Сега знаем, че CSS се използва за промяна на външния вид (стила). Смесването на двете се приема като лоша практика. Разделянето на визуалната част от структурата е това, към което трябва винаги да се стремим.

Урок 2: Част II

Начини на задаване на CSS

Чрез елемента `<style>` в раздела `<head>` на html документа

Този вид стил се нарича **Internal Style sheet** (вътрешен стил). Най-често се използва, когато искаме конкретна страница от нашия уеб сайт да бъде с уникален стил. Начина, по който дефинираме този вътрешен за страницата стил е като използваме елемента `<style>` в секцията `<head>` на нашата уеб страница.

Структура на style елемента

Преди да разгледаме един пример, нека се запознаем със структурата на *style* елемента. Този елемент идва под формата на двойка тагове – отварящ и затварящ. В отварящия таг (`<style>`) трябва да поставим един **задължителен** атрибут и това е **type**. Атрибута **type** служи за идентифициране на съдържанието между отварящия `<style>` и затварящия `</style>` таг. За сега стойността, която **type** може да приеме е само една – **text/css**. Визуално, всичко това може да бъде представено така:

```
<head>
```

```
.
```

```

<style type="css/text">
.
.
</style>
</head>

```

Кода между отварящия **<style>** и затварящия **</style>** таг се третира от брауъра като **CSS**. Остава да обясня още едно нещо преди да приложим всичко в нагледен пример. **Как да избира елемента, който искам да стилизирам?** При прилагането на стил чрез атрибута *style* видяхме, че е възможно да променим визията на даден xHTML елемент като директно въведем желаните CSS декларации. При този метод обаче не е така. Тук, ако директно въведем CSS декларациите нищо няма да се промени, защото брауъра няма как да разбере върху кой точно елемент да ги приложи. Затова при този метод, и при използването на външен Style sheet, CSS придобива малко по различна структура от тази разгледана в урока "Въведение в CSS".



Като цяло, можем да разделим структурата на 3 части:

- 1) **Селектор (selector)**. Това е мястото, в което избираме елемента, върху който искаме да приложим желания стил. Думата селектор е твърде общо понятие. Селектор може да бъде всеки xHTML елемент на нашата страница. Селектор може да бъде всяко id или class. След като разгледаме един пример, смятам че това ще ти се изясни.
- 2) Този вид скоби служат за ограничение на количеството CSS, което ще бъде приложено върху селектора. С други думи, кода поставен между отварящата { и затваряща скоба } служи за стилизиране на елемента избран като селектор.
- 3) Общото наименование на двойката **свойство : стойност** се нарича **декларация**. В урока "Въведение в CSS" дадох подробно описание на тази част затова няма да го описвам отново тук. Ако в момента това ти се струва малко неясно, по-добре прочети първо този урок.

Пример:

Стъпка 1: Ще създадем нова уеб страница като използваме *Strict Doctype*. Също така ще добавим едно h2 заглавие и един текстов параграф.

```

1. <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
2. "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
3. <html xmlns="http://www.w3.org/1999/xhtml">
4.     <head>
5.         <meta http-equiv="Content-Type" content="text/html;
6.             charset=utf-8" />
7.         <title>YouDevelop.net - Прилагане на CSS чрез
8.         елемента style</title>

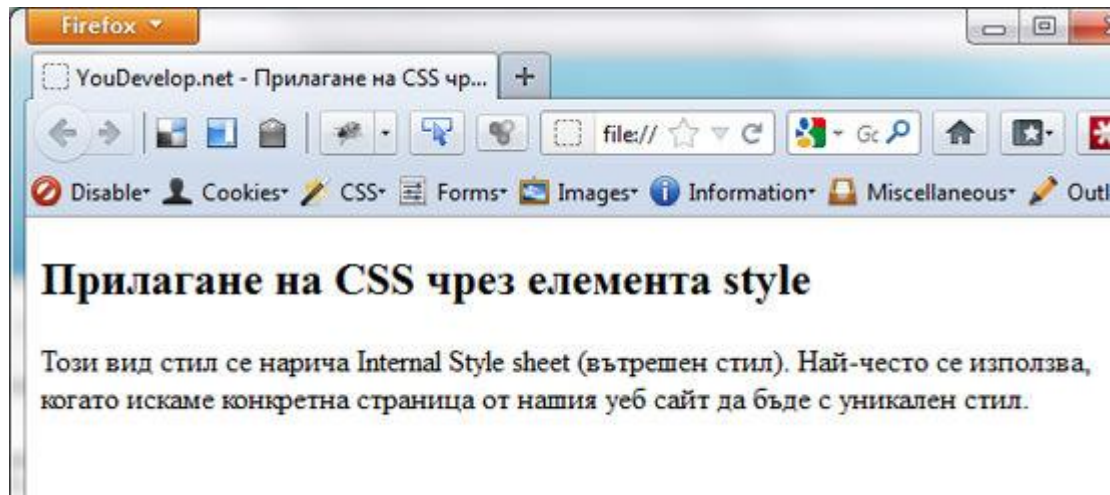
```

```

9.     </head>
10.    <body>
11.        <h2>Прилагане на CSS чрез елемента style</h2>
12.        <p>Този вид стил се нарича Internal Style sheet
13.        (вътрешен стил). Най-често се използва, когато искаме
14.        конкретна страница от нашия уеб сайт да бъде с
15.        уникален стил.</p>
16.    </body>
17. </html>

```

След като запазим документа и го отворим с браузър, страницата ще изглежда по следния начин:



Стъпка 2: По подразбиране цвета на заглавието и текста е черен. За да упражним наученото до сега нека сменим цвета на заглавието на оранжев. Също така нека наблегнем на думата "Internal Style sheet" като я удебелим и също ѝ променим цвета на оранжев. В `<head>` раздела на документа, веднага след `<title>` постави новият елемент `<style>`.

```

1. <title>YouDevelop.net - Прилагане на CSS чрез
2. елемента style</title>
3. <style type="text/css">
4.
5. </style>

```

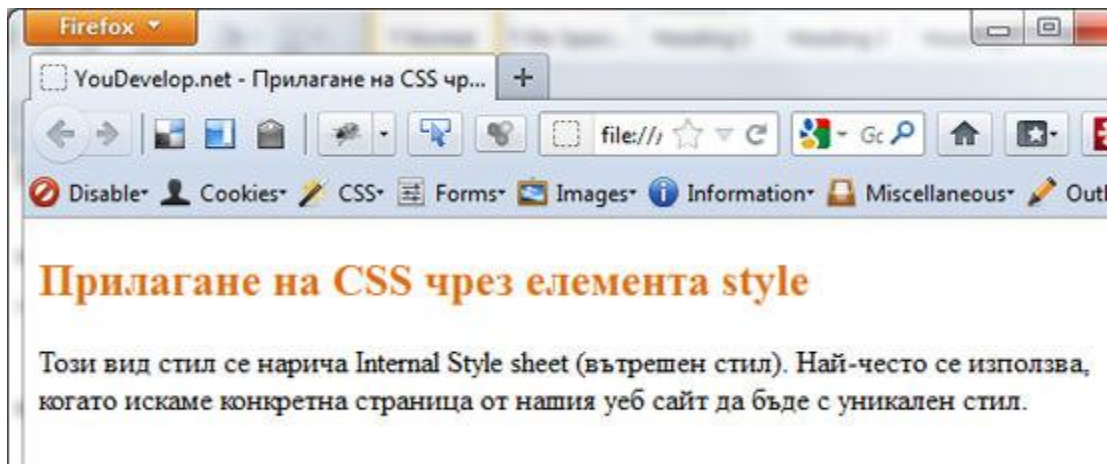
Стъпка 3: Първо ще променим цвета на заглавието. Между отварящия `<style>` и затварящия `</style>` таг въведи следния код:

```

1. h2 {
2.     color:#dd7319;
3. }

```

Запази промените и виж страницата в браузър:



Цвета на заглавието се промени успешно. В този случай **h2 е селектор** т.е. избираме (селектираме) **всички h2 заглавия**, които се намират в конкретната страница. Ако сега добавиш още едно h2 заглавие след текста, то неговия цвят също ще бъде променен. Всички декларации, които са между { и } ще бъдат **приложени върху селектора**. В случая имаме само една.

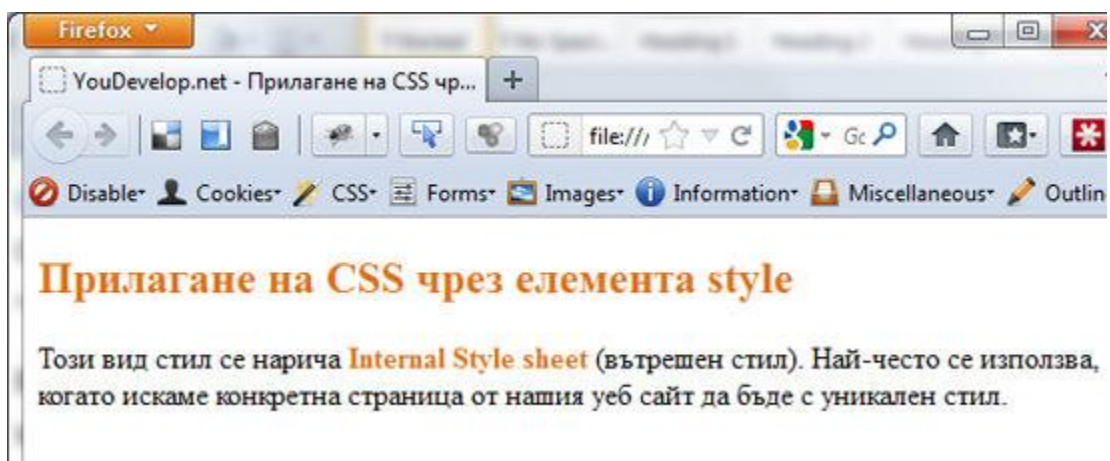
Стъпка 3: Сега нека удебелим и променим цвета на текста "Internal Style sheet", намиращ се в параграфа. Елемента **strong** служи за удебеляване на шрифта. Нека го приложим. Промени кода по следния начин:

1. `<p>Този вид стил се нарича Internal Style sheet`
2. `(вътрешен стил). Най-често се използва, когато искаме конкретна`
3. `страница от нашия уеб сайт да бъде с уникален стил.</p>`

Ако сега запазиш промените и отвориш страницата в браузър ще видиш, че текста "Internal Style sheet" е удебелен, но цвета си остана черен. С помощта на CSS можем да го променим без никакви проблеми. След CSS кода, който имаме до сега, добави следното:

1. `strong {`
2. `color:#dd7319;`
3. `}`

Запази и виж резултата в браузъра.



Освен, че текста е удебелен, успяхме и да променим цвета му.

В този случай **strong** играе ролята на селектор. Както и преди, така и тук кода между { и } ще се приложи върху **всички strong елементи** намиращи се в страницата независимо от техния брой. Пробвай да оградиш още 1-2 думи със strong, за да видиш как работи. Общото между двата стила е, че използвахме HTML елементи като селектори. **Всеки HTML елемент, който се намира в <body> може да бъде използван като селектор в CSS.** Единственото, което трябва да запомниш е, че когато използваш HTML елемент като селектор **не трябва да добавяш** символите < и >, а директно пишеш името на елемента. Друго много важно нещо е, че когато използваш даден HTML елемент като селектор, **браузъра ще приложи CSS стил върху всеки елемент от този вид.**

Хитринка: Надявам се, че забележа общото между CSS кода, който прилагаме върху h2 заглавието и кода, който прилагаме върху strong. Да, точно така, той е един и същ. В случаите, когато използваш един и същи код, за да стилизираш няколко различни елемента (2 или повече) можеш да използваш един трик, който ще намали излишното дублиране на код. При такива обстоятелства можеш да приложиш един и същи код върху няколко селектора като ги обединиш на един ред и ги отделиш със запетайка. Ето как става това:

```
1. Преди
2. h2 {
3.     color:#dd7319;
4. }
5. strong {
6.     color:#dd7319;
7. }
8.
9. Обединени
10. h2, strong {
11.     color:#dd7319;
12. }
```

Ако провериш резултата в браузър ще видиш, че страницата изглежда по абсолютно същия начин, но всъщност намалихме кода наполовина. Това, което казваме на браузъра е “Намери всяко h2 и strong на страницата и приложи върху тях кода между { и }”. Надявам се, че урока до сега е ясен и лесен за следване. Ако имаш съмнения и въпроси, сподели ги след коментарите.

Обобщение

Разгледахме втория метод за прилагане на CSS в нашата страница. Този се различава коренно от първия и според мен е много по-ефективен. Нека разгледаме предимствата и недостатъците му.

Предимства

- CSS от този метод се прилага само върху елементите, които са част от настоящата страница. Ако твоя уеб сайт съдържа няколко страници и искаш една от тях да има уникален стил, можеш свободно да използваш този метод за да я стилизираш.
- Няма нужда от създаването на допълнителен файл. Всичко, което трябва да направиш е да добавиш елемента <style> в раздела <head> на страницата.
- Разрешена е употребата на class и id като селектори. Тази част все още не сме я обхванали затова няма да навлизам в подробности. Разбира се, в някой от бъдещите уроци ще се запознаем със същността на class и id и тяхното приложение.

- Вътрешния стил (Internal Style sheet) има по-висок преоритет от стила приложен чрез външен стил (External Style sheet). Повече за това в някой от бъдещите уроци.
- Отделяне на структурата от визуалната част. В предишния метод въвеждахме CSS директно в xHTML елементите. Тук тези два компонента са разделени. Така много по-лесно можем да направим промени в кода като вероятноста да оплескаме нещата е по-малка.

Недостатъци

- Този метод е ефективен само на страницата, в която е приложен. Въпреки, че това го изброих като предимство понякога може да бъде и недостатък, защото CSS се използва само от настоящата страница. Ако искаме друга страница да бъде със същия стил, трябва да копираме кода (което довежда до излишно дублиране) или да го сложим във външен файл (External Style sheet).
- Увеличава времето за зареждане на страницата. Всеки път, когато отворим страница, браузъра трябва отново и отново да премине през всеки селектор и декларация което малко или много отнема време.
- Понякога е трудно да се прецени на пръв поглед кой стил върху кой точно елемент ще бъде приложен. Особено за начинаещите в CSS това може да бъде малко объркващо, но обикновено с течение на времето (и натрупването на знания) нещата се изглаждат.

Урок 2: Част III Начини на задаване на CSS Чрез външен файл (External Style sheet)

Този метод се нарича **External Style sheet** (външен стил). Това е **най-широко разпространения** метод за прилагане на CSS към уеб страница.

Начина по който дефинираме този стил е като **създадем нов документ**, който запазваме с **разширение .css**. Този документ съдържа единствено CSS стила, който искаме да използваме. След това посредством **елемента <link>**, който поставяме в секцията <head> на нашата уеб страница, “прикачваме” новосъздадения CSS файл.

Това е кратко описание на процеса. По-надолу в урока ще разгледаме един пример, който обхваща всичко това.

Структура на елемента link

Успеха на този метод се осланя изцяло на елемента <link>. Този елемент ни позволява да “свържем” настоящия документ с външен файл, чрез който можем да стилизираме нашата страница.

Елемента <link> е от групата на “празните” елементи. Това означава, че елемента се затваря като добавим дясно наклонена черта / преди края на отварящия таг.

Ако искаш да научиш повече за видовете елементи, в урока “Разлики между HTML и xHTML” говорих повече за това.

Също така **елемента <link> съдържа няколко атрибута, които не са задължителни**, но използването им е силно препоръчително, за да бъде кристално ясно на браузъра, че файла който прикрепяме е CSS.

```

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

```

Структурата на `<link>` може да бъде разделена на няколко части:

1. **Атрибута href:** Стойността на този атрибут определя **мястото до ресурса**, включително неговото име, последвано от разширението `.css`. Когато файла се намира в същата папка, в която е и `html` документа, тогава като стойност на `href` изписваме единствено името на файла и неговото разширение. От снимката виждаш, че стойността на `href` е `"style.css"`, което означава че файла се казва `style`. Когато файла се намира в отделна папка, тогава задаваме пътя до файла включително и имената на всички папки, през които преминаваме. Например, ако файла се намира в папка `style`, то тогава стойността на `href` ще бъде `"style/style.css"`. Ако файла се намира в директория, която е едно ниво по-високо от директорията, в която се намира `html` документа. Тогава стойността на `href` ще бъде `"../style.css"`, където `"../"` означава, че се изкачваме с една директория по-високо. Ако файла се намира две директории по-високо, тогава се изкачваме два пъти `"../style.css"` и т.н. Когато файла се намира някъде в интернет, тогава като стойност на `href` можем да зададем пълния адрес до този файл, например `http://www.example.com/style.css`
2. **Атрибута rel:** Този атрибут определя връзката на файла с документа, от който го извикваме. **rel** може да приема различни стойности, но тази която в момента ни интересува и която ще използваме почти винаги, е **stylesheet**.
3. **Атрибута type:** Както в урока Прилагане на CSS чрез елемента `<style>`, така и тук, **стойността на type служи за идентификация на съдържанието**. Може да приема различни стойности, но тази която ще използваме е **text/css**.
4. Както казах по-рано, елемента `<link>` е "празен" елемент, което означава че се затваря като добавим дясно наклонена черта / преди края на отварящия таг. Важно е да отбележа, че **няма ограничение на броя <link> елементи, които можем да използваме в нашата страница**. Всъщност, всичко зависи от сложността на уеб сайта, но прекалено многото понякога затруднява поддръжката им.

Съвет: "Винаги започвай с един външен файл и да добавяй втори или следващ, тогава когато почувстваш, че наистина има нужда от тях".

Пример:

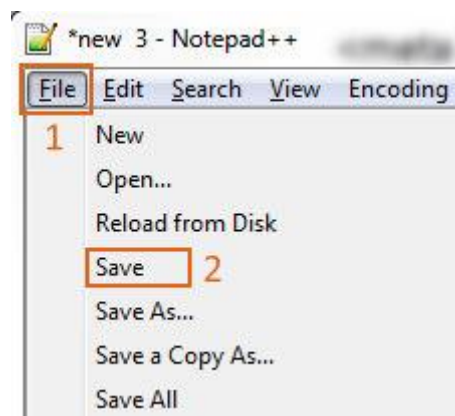
Стъпка 1: Ще създадем нова уеб страница като използваме *Strict Doctype*. Също така ще добавим едно `h2` заглавие и един текстов параграф.

```

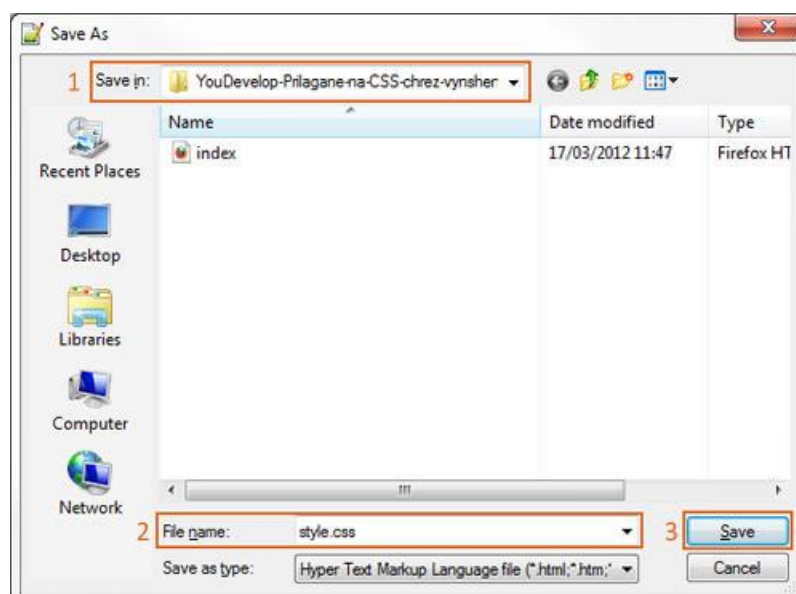
1. <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
2. "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
3. <html xmlns="http://www.w3.org/1999/xhtml">
4.     <head>
5.         <meta http-equiv="Content-Type" content="text/html;
6.             charset=utf-8" />
7.         <title>YouDevelop.net - Прилагане на CSS чрез външен
8.             файл</title>
9.         <link href="style.css" rel="stylesheet" type="text/css" />
10.    </head>
11.    <body>
12.        <h2>Прилагане на CSS чрез външен файл</h2>
13.        <p>Този метод се нарича External Style sheet
14.        (външен стил). Това е най-широко разпространения
15.        метод за прилагане на CSS към уеб страница.</p>
16.    </body>
17.</html>

```

Стъпка 2: Сега нека създадем CSS файла, който ще използваме за стилизиране на нашата страница. Отвори Notepad++ (или редактора, който използваш) и създай нов документ. В стъпка 6 ще въведем нужния CSS код. За сега нека просто запазим файла в правилния формат. От менюто File (1) избери Save (2)



Ще се отвори нов прозорец, в който трябва да избереш **мястото** (1), където ще запазиш файла и **неговото име** (2).



Препоръчвам ти да запазиш файла в същата папка, в която се намира html документа. В полето за **име** (2) въведи име по твое предпочитание. Не забравяй ведната след името да добавиш и разширението на файла **.css**. Ако не го направиш програмата може да избере грешен формат, който няма да ти бъде от полза. В моя пример името на файла ще бъде "style.css" Натисни **бутона Save** (3), за да запазиш документа.

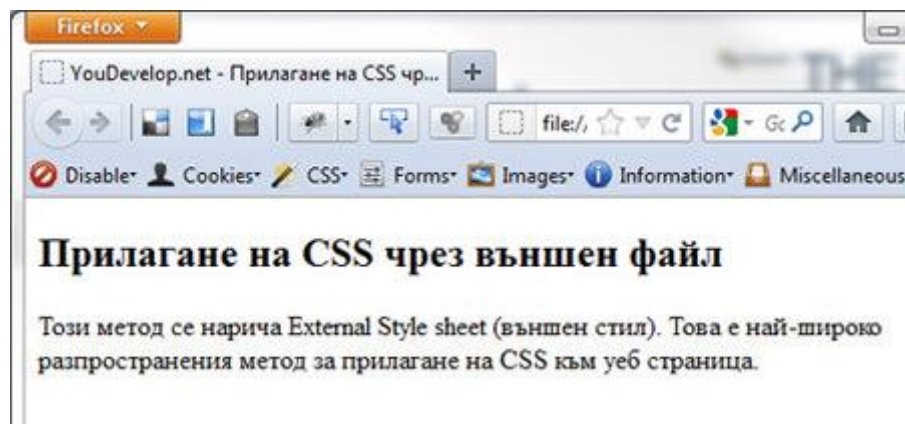
Забележка: Не забравяй да запазиш файла под UTF-8 without BOM енкодинг. Можеш да провериш дали този енкодинг е избран от менюто Encoding. Ако не е избран, избери го преди да запазиш файла.

Стъпка 3: След като вече имаме файла, нека го свържем с html документа. В <head> секцията добави <link> елемента, който разгледахме по-горе. Кода ще придобие следния вид:

1. <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
2. <title>YouDevelop.net - Прилагане на CSS чрез външен файл</title>
3. <link href="style.css" rel="stylesheet" type="text/css" />
4. </head>

Стъпка 4: Скелета на уеб страницата е готов, остава да добавим стила.

Ако отворим страницата в момента, ще видим че цвета на заглавието и текста е черен. Въпреки че изглежда достатъчно добре, все пак целта ни е да упражним CSS, затова нека го променим.



Ще направим заглавието оранжево. Също така ще наблегнем на думите "External Style sheet" и "най-широко разпространения метод" като ги удебелим и променим цвета им също на оранжев.

Стъпка 5: Първо ще променим xHTML кода като добавим нужните елементи. За да удебелим двете набелязани думи, трябва просто да ги оградим с елемента .

1. <h2>Прилагане на CSS чрез външен файл</h2>
2. <p>Този метод се нарича External Style sheet
3. (външен стил). Това е най-широко разпространения
4. метод за прилагане на CSS към уеб страница.</p>

Общо взето нищо ново. Бърз преглед на страницата ни показва, че всичко работи нормално.

Стъпка 6: Сега идва и реда на CSS. Отвори новосъздадения файл style.css и въведи следния код:

```
1. h2 {
2. color:#dd7319;
3. }
4. strong {
5. color:#dd7319;
6. }
```

Селекторите и декларациите при този метод се **дефинират по абсолютно същия начин**, който разгледахме в урока Прилагане на CSS чрез елемента style.

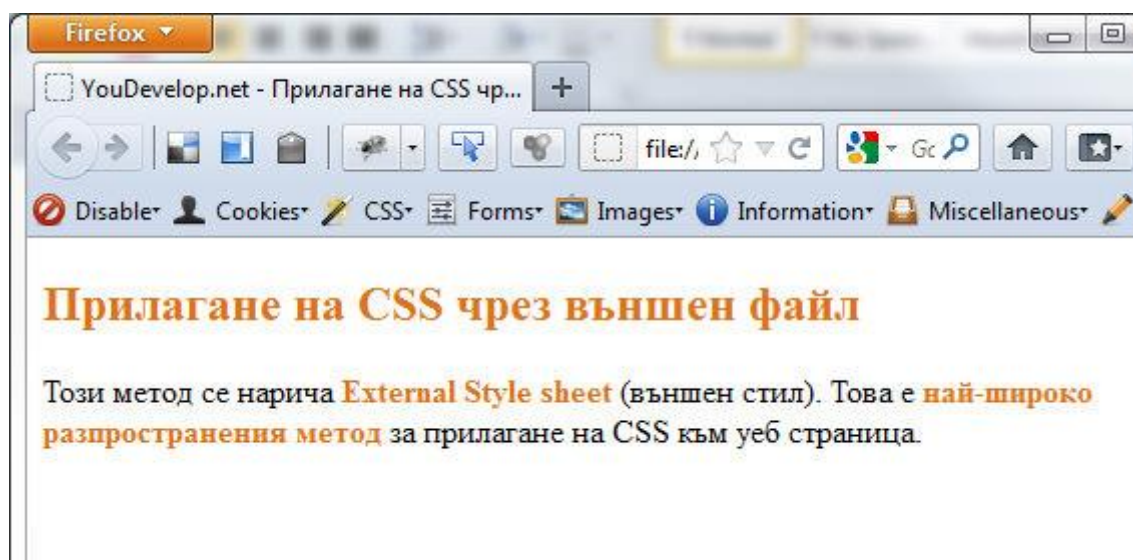
Правилата са напълно идентични. Всичко което важи за предишния метод, важи и тук. Разликата обаче е, че тук CSS се намира във външен файл, докато при предишния метод CSS кода беше част от самата страница.

Спомняш ли си за хитринката, която ти споделих в предишния метод? Е, тя може без никакви проблеми да бъде приложена и сега (тъй като стила приложен върху двата селектора е един и същ). След комбинирането на двата стила, кода ще придобие следния вид:

```
1. h2, strong {
2. color:#dd7319;
3. }
```

Стъпка 7: Всичко до тук изглежда логично и добре подредено, но нека видим дали работи.

Запази промените и отвори страницата в браузър. Ако си следвал/а внимателно всички стъпки до тук, страницата трябва да изглежда така:



Въпреки че променихме само малка част от стила, успяхме безпроблемно да “прикачим” външен стил (External Style sheet) към нашата страница. Надявам се, че намиращ урока лесен за следване и достатъчно ясен. Ако имаш съмнения или затруднения, или искаш да споделиш своя коментар, направи го след урока.

Обобщение

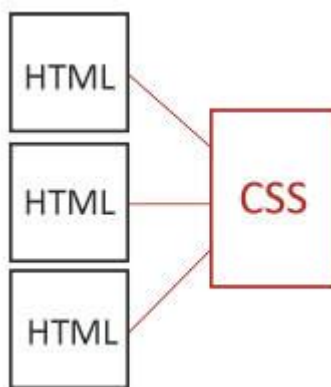
В този урок разгледахме и третия метод за прилагане на CSS към веб страница. Лично аз винаги го предпочитам пред останалите два, защото е много по-ефикасен и спестява време.

Недостатъци

- **Създаване на допълнителен файл.** Докато в предишните два метода CSS бе част от html документа, тук се нуждаем от втори файл, в който да съхраняваме стила на нашата страница;
- Външния файл може да увеличи времето за зареждане на страницата, ако е прекалено голям. Това най-вече се отнася за първото зареждане на страницата;
- Ако не контролираме количеството CSS код, който добавяме в този файл, той може да нарасне много бързо, което допълнително усложнява неговата поддръжка. Не казвам, че трябва да ограничаваме кода до 100 реда, а просто да бъдем внимателни с това колко всъщност код използваме в нашата страница. Например, ако 50 реда от нашия код е предназначен за старата версия на сайта и вече не се използва, то много по-добре би било, ако въобще го няма. Кода трябва да бъде чист и спретнат като оставяме само това, от което наистина се нуждаем;
- За начинаещите в CSS този метод може би в началото ще бъде малко дезориентиращ, тъй като селекторите и действителните елементи са на две различни места. Обикновено, с течение на времето (и естествено с достатъчно практика) нещата се изглаждат.

Предимства

- **Спестяване на време.** Едно от правилата при създаването на веб сайт е той следва определена последователност. Така посетителя има усещането, че все още се намира в същия сайт, независимо от страницата, която разглежда в момента. Един от начините да постигнем тази съгласуваност между страниците е като използваме един и същи стил. Отделянето на CSS кода в отделен файл ни предоставя тази възможност, защото можем да го приложим в няколко страници наведнъж. Това става ясно от следващата снимка.



Както виждаш, чрез използването на елемента <link> лесно можем да “свържем” няколко страници с един CSS документ.

- **Лесен за обновяване.** Представи си, че имаш 5 страници и искаш да промениш стила на даден елемент, който се намира във всяка от тях. Използвайки този метод това може да стане изключително бързо, защото промяната се прави само на едно място, но тя ще повлияе навсякъде.
- **Отделяне на структурата от визуалната част.** Както вече знаеш, HTML се използва за изграждане на структурата на нашата страница. CSS от своя страна въздейства върху стила ѝ. Отделянето на двете се счита за добра практика.

- **Намаля риска от грешки.** Отделянето на CSS кода в собствен файл до голяма степен ни подсказва, че съдържанието на този файл е CSS. Това ни прави по-внимателни при изписването на селекторите и декларациите, защото всеки друг код не би работил.

Урок 3: I част

Усвояване на CSS background

Усвояване на CSS background (част 1)

CSS е известен с богатото си разнообразие от свойства, чрез които изключително лесно можем да подобрим стила на нашата уеб страница.

Предполагам, че си попадал/а на уеб страница, която използва страхотна снимка за фон. В един сайт снимката е пейзаж, който ти дава чувството на свобода, в друг пък, текстура която придава определен стил.

Дали ще бъде определен цвят или снимка, всичко това разбира се зависи от сайта и неговите посетители, но едно може да бъде казано за всички тях: *“Правилният избор на фон обогатява изживяването в страницата и оставя у посетителя спомен, който трае дълго след като я напусне”*.

В следващите два урока ще ти помогна да овладееш CSS **свойството background** и ще се постарая да разясня всичко, което трябва да знаеш за него.

Свойството **background** е разделено на няколко части. Те са:

- background-color
- background-image
- background-repeat
- background-position
- background-attachment
- background (съкратено)

I. background-color

Това свойство определя цвета, който ще бъде използван за фон на избрания елемент.

За фон (*background*) на елемент се счита пространството, което се обхваща от ширината и височината на избрания елемент (независимо дали тези размери са фиксирани чрез ширина (*width*) и височина (*height*) или се определят от съдържанието на елемента).

Фона също включва и пространството обхванато от *padding* и *border* (повече за тях в бъдещи уроци).

Ако елемента **няма размери** (фиксиран или зависещ от съдържанието) то **цвета няма да бъде видим**. Също така, ако даден елемент съдържа други елементи, върху всички от които има проложен *float* (повече за това CSS свойство в някой от бъдещите уроци), то фона също няма да бъде видим, защото височината на този елемент е 0. Решение в този случай е добавянето на *clear* в края на тази група от елементи.

Ако искаш да промениш **фона на цялата страница**, то тогава трябва да приложиш **background-color** върху **<body>**.

background-color може да приеме 3 стойности:

1) background-color: { color | transparent | inherit };

- **color** - Стойността тук може да бъде всеки валиден CSS цвят или ключова дума. Това може да бъдат:

- **HEX стойности** - Специален шестнадесетичен rgb (red, green, blue) цветови код (на английски hexadecimal color). Например #dd7319;
- **RGB стойности** - Специален десетичен rgb цветови код. Например rgb(255,0,0);
- **Ключови думи**, които на английски значат определен цвят. Например "red", "green", "blue" и други.
- **transparent** - Това е **стойността по подразбиране**. Ако не зададем никаква друга стойност на *background-color*, то неговата стойност ще бъде *transparent*. *Transparent* означава, че фона на конкретния елемент ще бъде прозрачен т.е. ако зад избрания елемент има друг елемент, то фона на втория елемент ще бъде видим.
- **inherit** - Когато приложим *inherit* върху определен елемент, това означава че стойността на *background-color* ще бъде същата като стойността приложена върху родителския елемент т.е. стойността ще бъде унаследена. Това свойство се използва рядко.

Пример

Стъпка 1: Създай нов XHTML документ, който да използва *Strict Doctype* и да съдържа едно h2 заглавие и кратък текстов параграф. Също така нека удебелим 2-3 думи от този текстов параграф чрез *strong*.

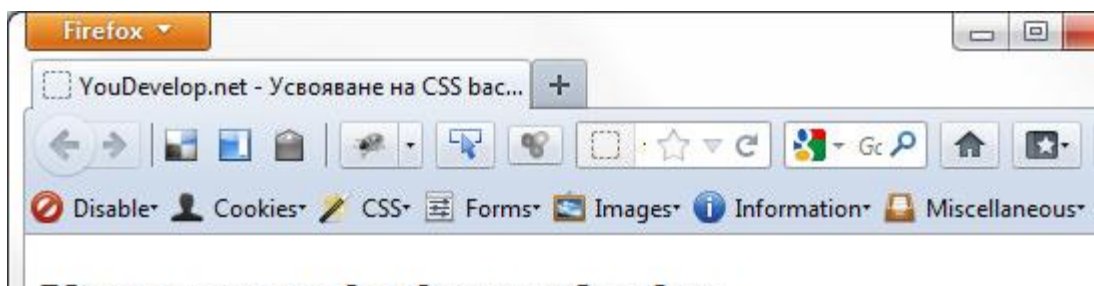
В това упражнение ще променим фона на страница, h2 заглавието и на текста, който е удебелен.

```

1. <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
2. "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
3. <html xmlns="http://www.w3.org/1999/xhtml">
4.   <head>
5.     <meta http-equiv="Content-Type" content="text/html;
6.       charset=utf-8" />
7.     <title>YouDevelop.net - Усвояване на CSS background -
8.     background-color</title>
9.     <link href="style.css" rel="stylesheet"
10.        type="text/css" />
11.   </head>
12.   <body>
13.     <h2>Усвояване на background-color</h2>
14.     <p>Това свойство определя цвета, който ще бъде
15.     използван за <strong>фон на избрания елемент</strong>.
16.     </p>
17.   </body>
18. </html>

```

В браузъра ще видим следния резултат:

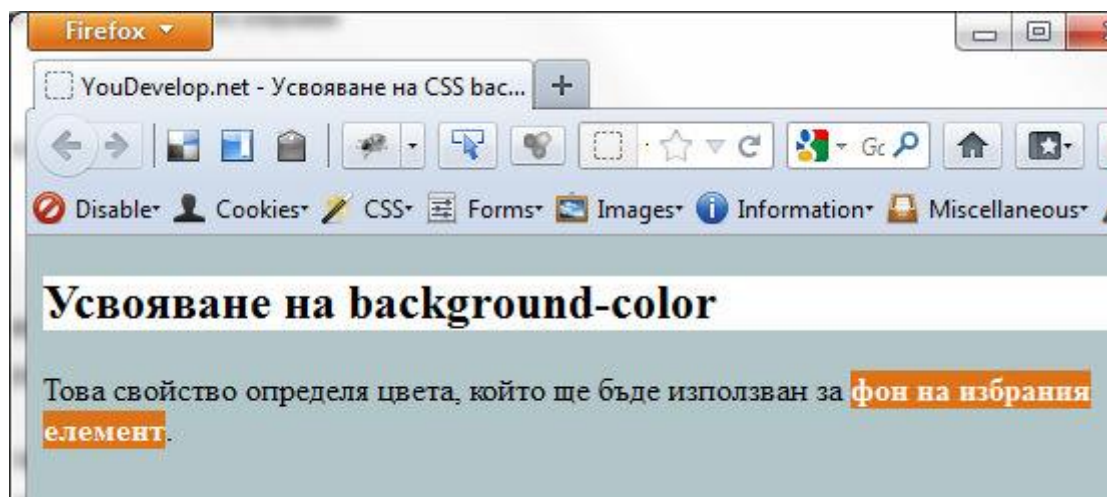


Стъпка 2: В кода от предишната стъпка можеш да забележиш, че използвам външен файл за съхранение на CSS кода. Тази техника я разгледахме в урока "Прилагане на CSS чрез външен файл".

Създай нов документ, който ще използваш за CSS. Запази го при html файла под името "style.css". Сега го отвори и въведи следния код:

```
1. body {  
2.   background-color:#b1c6c8;  
3. }  
4. h2 {  
5.   background-color:white;  
6. }  
7. strong {  
8.   background-color:rgb(221,115,25);  
9.   color:#fefefe;  
10.  }
```

Стъпка 3: Запази промените и отвори страницата в браузър.



Резултата не е от най-красивите, но определено ни дава добра представа за работата на *background-color*.

В този пример съм използвал трите възможности за задаване на цвят.

- **Фона на body** е шестнадесетичен rgb цветови код, който започва с #, последван от 6 символа представляващи цвета. Това могат да бъдат числата от 0 до 9 и буквите от "a" до "f".
- **Фона на h2 заглавието** е английската дума "white", която отговаря на "бял".

- **Фона на strong** е зададен чрез десетичен RGB код, който започва с **rgb(**, последван от 3 стойности за всеки цвят разделени със запетайка. Цветовете се представят чрез цифрови стойности в обхвата от 0 до 255. Накрая се затваря скобата).
- **Фона на p** е *transparent*. Това е стойността по подразбиране и няма нужда от повторното ѝ задаване. Когато даден елемент има *background-color:transparent;*, елемента който е зад него ще се вижда. В случая се вижда фона на страницата.

Подробно разгледахме как работи *background-color*. Надявам се, че всичко до тук е разбираемо и ясно. Ако имаш въпроси, сподели ги след урока.

II. background-image

Това свойство **определя снимката**, която ще бъде използвана за фон на избрания елемент.

Правилата, които важат за прилагане на фон чрез *background-color*, важат и тук. Прочети ги отново в предишната точка.

По подразбиране снимката използвана за фон започва от горния ляв ъгъл на страницата и се повтаря хоризонтално (x) и вертикално (y), докато стигне долния десен ъгъл на страницата. Не се безпокой, всичко това може да бъде контролирано чрез CSS. В следващия урок ще опиша подробно как става това.

Ако искаш да използваш снимка за **фон на цялата страница**, то тя най-често се **прилага върху <body>**.

background-image може да приеме 3 стойности:

1) `background-image: { uri | none | inherit };`

- **uri** - Като стойност тук задаваме пътя до снимката, която ще използваме за фон. Имаме няколко възможности:
 - Когато файла се намира в същата папка, в която е css документа, тогава като стойност на **uri** изписваме единствено името на файла и неговото разширение.
 - Когато файла се намира в отделна папка, тогава задаваме пътя до файла включително и имената на всички папки, през които преминаваме. Например, ако файла се намира в папка *images*, то тогава стойността ще бъде *"images/snimka.jpg"*.
 - Ако файла се намира в директория, която е едно ниво по-високо от директорията, в която се намира css документа. Тогава стойността ще бъде *"../snimka.jpg"*, където *".."* означава, че се изкачваме с една директория по-високо. Ако файла се намира две директории по-високо, тогава се изкачваме два пъти *"../snimka.jpg"* и т.н.
 - Когато файла се намира някъде в интернет, тогава като стойност можем да зададем пълния адрес то този файл, например *"http://www.example.com/snimka.jpg"*.

В урока "Работа с изображения" говорих за поддържаните файлови формати в Интернет. Това важи и тук. Опитай се да използваш снимки, които са в един от изброените формати.

Начина, по който използваме `background-image` е:

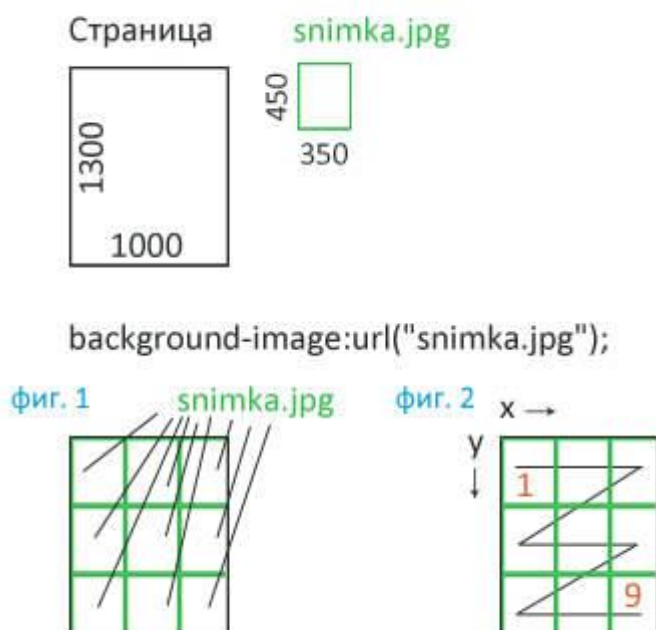
```
background-image:url('snimka.jpg');  
background-image:url("snimka.jpg");
```

background-image: последвано от "url(", последвано от еденична (') или двойна (") кавичка, последвано от пътя до снимката, последвано от еденична (') или двойна (") кавичка и накрая затваряме скобата). Двете кавички трябва да бъдат еднакви т.е. ако използваш еденична в началото, трябва да използваш еденична и в края.

- **none** – Това е **стойността по подразбиране**. Ако не зададем никаква друга стойности на *background-image*, то неговата стойност ще бъде none. От друга страна, изричното поставяне на "none" ще **предотврати** показването на *background-image*.
- **inherit** – Тази стойност ще позволи на елемента да наследи стойността на *background-image* от родителския елемент. Този подход обикновено не се използва, защото фона на елемента и неговия родител ще се застъпят.

Как работи background-image?

Така изглежда визуално работата на *background-image*:



Нека предположим, че страницата ни е с размери 1300px (височина) на 1000px (ширина), а снимката, която ще използваме за фон, е с размери 450px (височина) на 350px (ширина). Виждаме, че размерите на снимката са значително по-малки от тези на страницата, затова когато я приложим като фон, тя ще се повтори многократно докато изпълни ширината и височина на цялата страница (виж фиг. 1).

Във фиг. 2 можеш да видиш реда, по който се прилага *background-image*. Снимката започва от горния ляв ъгъл (1) на страницата и свършва в долния десен ъгъл (9).

Ако искаш снимката на твоята страница да не се повтаря, имаш няколко варианта, един от които е да използваш снимка с размери по-големи от тези на страницата. Така обаче част от снимката може да остане невидима за посетителя.

Пример

Стъпка 1: Ще използваме същата страница, която създадохме по-рано. Единствената промяна ще бъде в CSS кода. Създай нов html и css документ, ако не искаш да променяш предишните. Избери снимка, която искаш да приложиш за фон. Аз ще работя с ето тази:

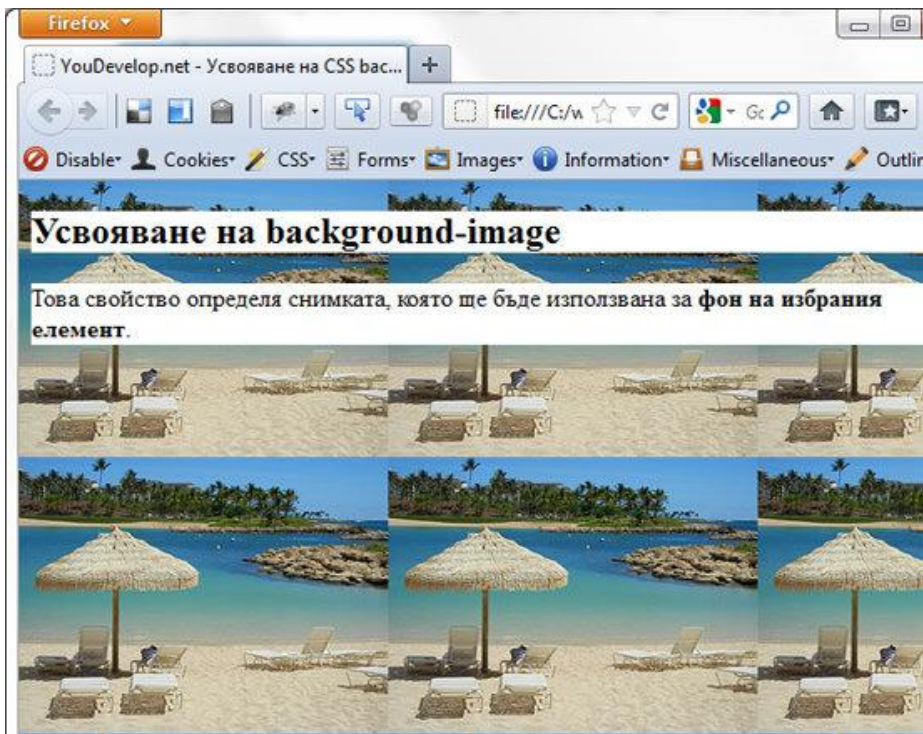


Постави снимката в **същата папка**, в която се намират html и css документите.

Стъпка 2: Отвори css файла и въведи следния код:

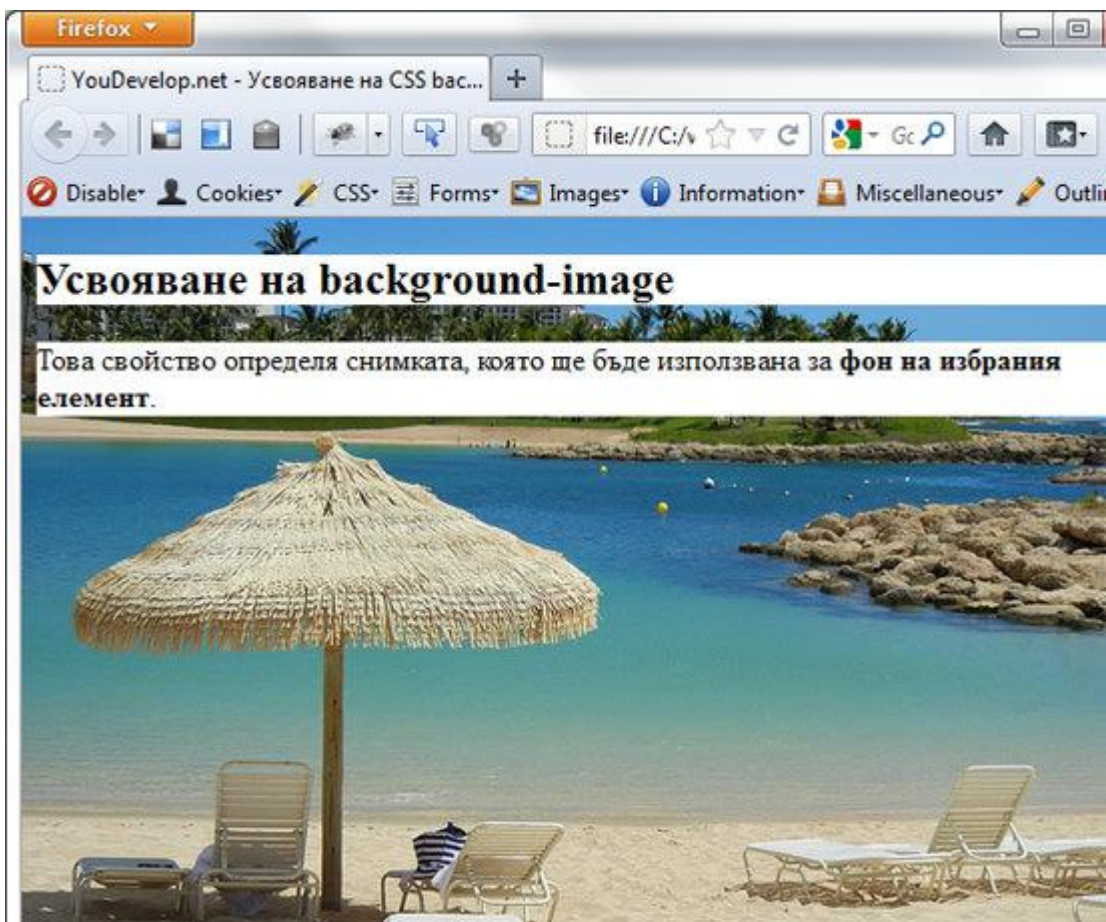
```
1. body {  
2.   background-image:url("snimka.jpg");  
3. }  
4. h2, p {  
5.   background-color:white;  
6. }
```

Стъпка 3: Запази промените и отвори страницата с браузър.



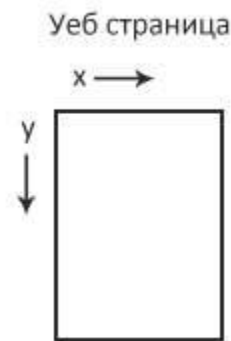
Резултата не е от най-красивите, но ясно можеш да видиш как функционира *background-image*.

Ако увеличим размера на снимката дотолкова, че да стане по-голяма от размера на страницата, тогава резултата ще бъде значително по-качествен.



III. background-repeat

Това свойство контролира дали изображението поставено чрез *background-image* ще се повтаря и ако се повтаря, то ще определя посоките на повтаряне (хоризонтално - **x**, вертикално - **y** или и двете (виж следващата снимка)).



background-repeat може да приеме 5 стойности:

```
background-repeat: { repeat|repeat-x|repeat-y|no-repeat|inherit };
```

- **repeat** - Това е стойността по подразбиране. Изображението използвано за фон **ще се повтаря** по хоризонтала (x) и вертикала (y) от горният ляв ъгъл на елемента до долният десен ъгъл, докато целия фон на елемента е напълно обхванат.
- **repeat-x** - Тази стойност **ограничава повтарянето само по хоризонтала (x)** в посока от ляво на дясно, докато фона на елемента е напълно обхванат по тази ос.
- **repeat-y** - Тази стойност е идентична на предишната, но тук **повторението е по вертикала (y)** в посока от горе надолу, докато фона на елемента е напълно обхванат по тази ос.
- **no-repeat** - Тази стойност **предотвратява повтарянето** и в двете посоки (вертикално и хоризонтално). В този случай изображението се показва само веднъж като мястото му се определя от координатите зададени чрез *background-position*. Ако изрично не зададем координати, изображението ще се покаже в горния ляв ъгъл на елемента.
- **inherit** - Тази стойност унаследява стойността приложена върху родителския елемент. Използва се много рядко.

Пример

Стъпка 1: Предлагам ти да използваме кода за прилагане на *background-image*, който използвахме в предишния урок.

Свали работните файлове към [предишния урок](#). След като ги разархивираш, в папката "*background-image*" ще намериш други две папки. Сега ще използваме файловете от папка "*primer-1*".

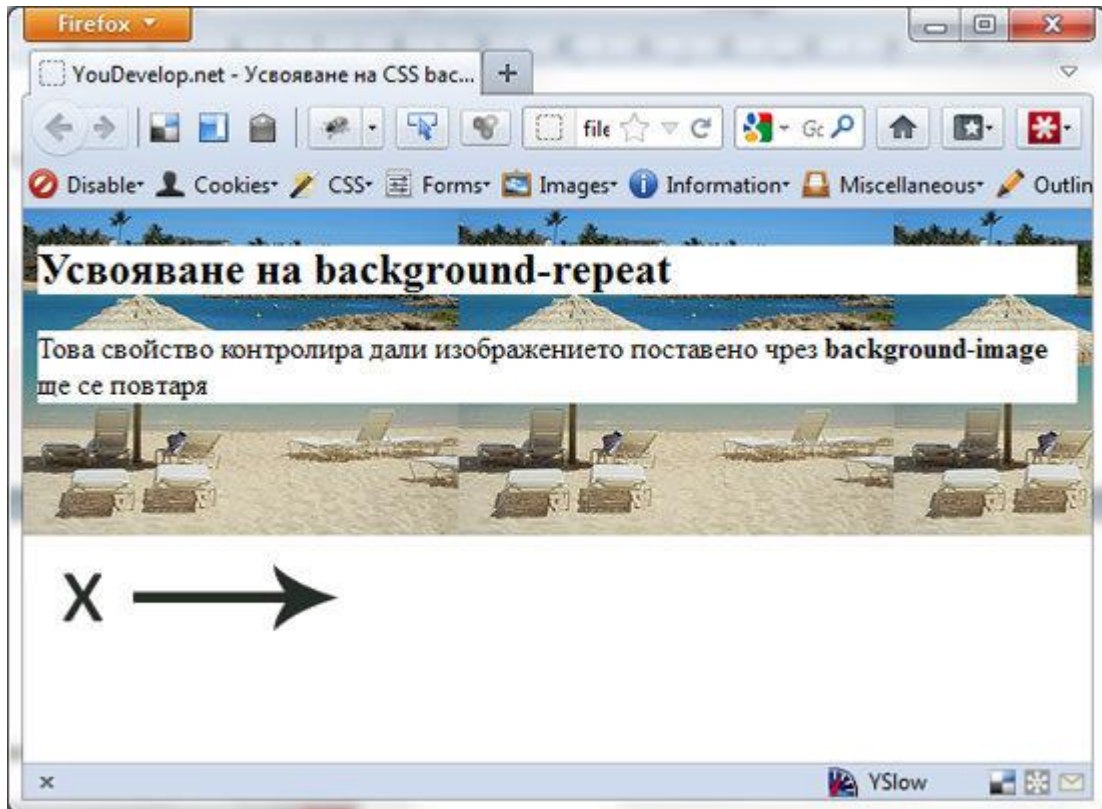
Няма значение дали ще ги копираш или ще работиш върху същите файлове. Отвори "*style.css*" и нека променим кода.

Стъпка 2: Вече знаем, че стойността **repeat** се използва по подразбиране, затова няма смисъл от повторното ѝ задаване. Дори да я използваме, резултата ще бъде абсолютно същия, **но пък ще сме добавили код от който всъщност нямаме никаква полза. Ще преминем директно към repeat-x.**

Стъпка 3: В кода, който имаме до сега, добави свойството *background-repeat* и му задай стойност *repeat-x*. Това ще накара изображението да се повтаря по хоризонталната ос (x).

```
1.     body {
2.         background-image:url("snimka.jpg");
3.         background-repeat:repeat-x;
4.     }
5.     h2, p {
6.         background-color:white;
7.     }
```

Запази промените и виж резултата в браузър.

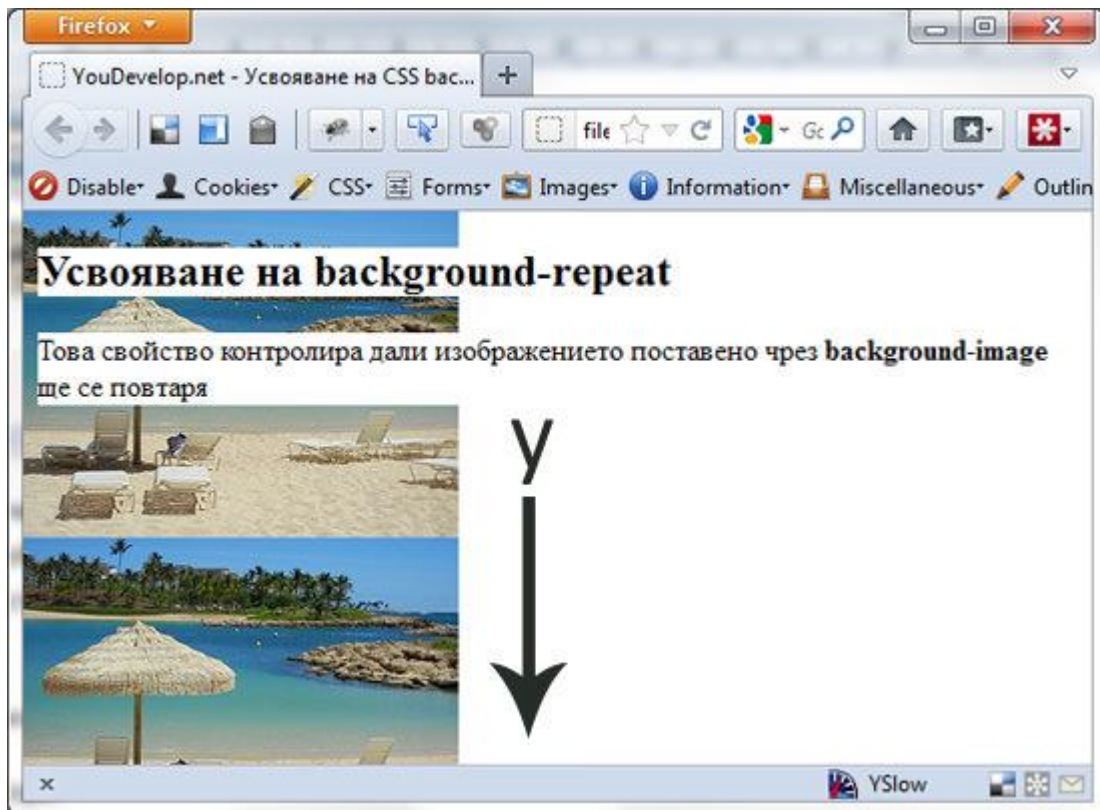


Изображението вече **не се повтаря в цялата страница, а само по хоризонтала**. В този случай, освен че имаме изображение за фон, можем да използваме и цвят, който ще бъде видим в останалата част от страницата.

Стъпка 4: **Видяхме как работи repeat-x. Сега нека пробваме repeat-y:**
Промени кода по-горе:

```
1. body {
2.     background-image:url("snimka.jpg");
3.     background-repeat:repeat-y;
4. }
5. ...
```

Запази промените и виж резултата в браузър.

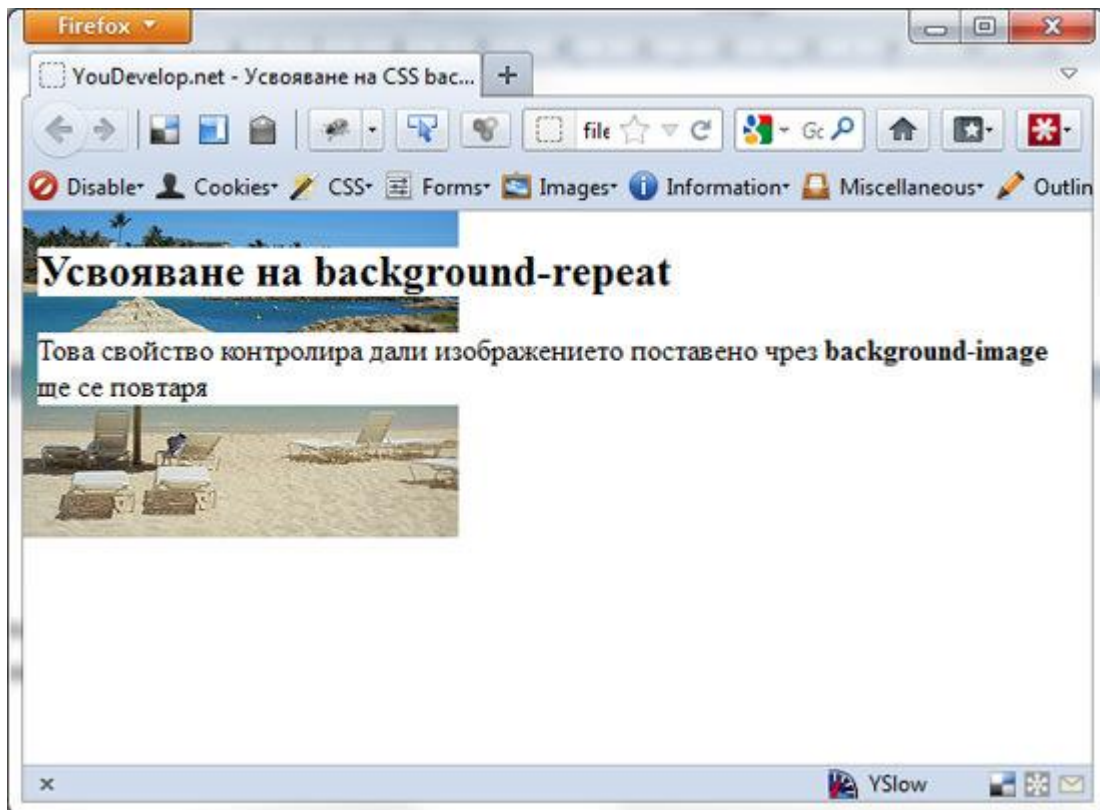


Резултата е наистина впечатляващ. Сега изображението се повтаря **само по вертикалната ос (y)**. Тук също можем да приложим *background-color*, който ще бъде видим в останалта част от страницата.

Стъпка 5: Сега идва ред да изпробваме работата на `no-repeat`. Промени кода по-горе като този път нека *background-repeat* приеме стойност `no-repeat`:

```
1. body {  
2. background-image:url("snimka.jpg");  
3. background-repeat:no-repeat;  
4. }  
5. ...
```

Резултата е точно това, което очаквахме.



Изображението **не се повтаря** нито по хоризонтала, нито по вертикала. Тъй като не сме определили неговите координати чрез [background-position](#) (това ще го направим след малко), по подразбиране изображението застава в горния ляв ъгъл на елемента (в случая това е страницата като цяло).

Разгледахме подробно свойството **background-repeat**. Надявам се, че всичко до тук е ясно и разбираемо. Ако имаш въпроси и коментари, сподели ги след урока.

IV. **background-position**

Това CSS свойство определя **мястото** на което ще бъде поставено изображението използвано за фон. След като зададем мястото му, можем да използваме предишното свойство ([background-repeat](#)), за да го повторим по вертикала и/или хоризонтала.

Ако не зададем никакви стойности на **background-position**, то изображението по подразбиране ще застане в горния ляв ъгъл на елемента (което отговаря на координатите 0,0).

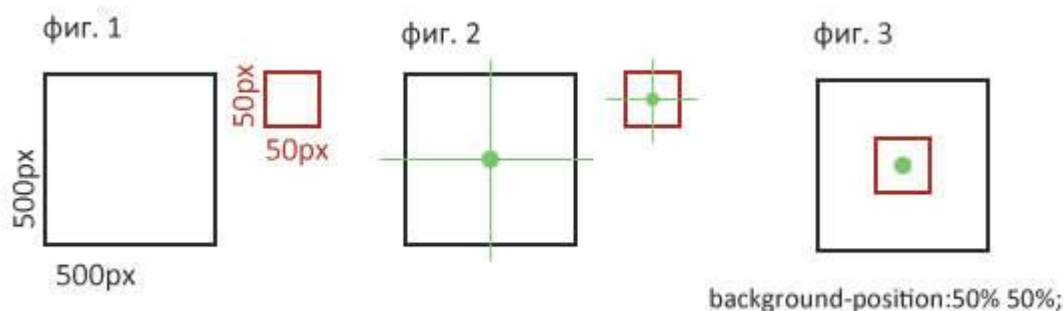
background-repeat може да приеме комбинация от няколко стойности:

`background-position: {{percentage|length|keywords}} (1 или 2 стойности) |inherit;`

- **percentage (процент)** – Можем да определим мястото, на което ще застане изображението като използваме проценти. Тези стойности могат да бъдат малко подвеждащи (особено за начинаещите), защото при тях се взема под внимание размерите на изображението и размерите на елемента, към който прилагаме *background-position*.

Пример: Имаме уеб страница, която е 500px широка и 500px висока. Изображението, което искаме да поставим за фон е с размери 50px ширина на 50px височина (фиг. 1). Когато искаме снимката за фон да бъде поставена на 50% от ширината и 50% от височината на елемента (фиг. 2), браузъра от своя страна изчислява къде се намира 50% от ширината и 50%

от височината на изображението (фиг. 2), след което поставя тази точка от изображението на определеното място в елемента.



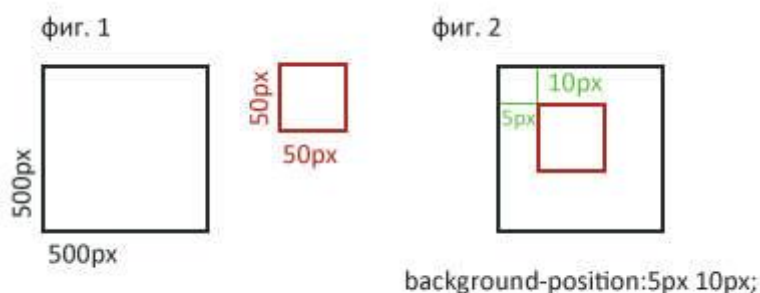
В горния случай изображението е центрирано перфектно.

Ако използваме само една процентна стойност, браузъра ще приеме като втора стойност ключовата дума "center".

Допустимо е използването на негативни стойности (например `background-position:-50% -10%;`)

- **length (дължина)** - Тук имаме възможност да използваме различни стойности (px, cm, mm, in и др), които определят мястото на изображението спрямо горния ляв ъгъл на елемента.

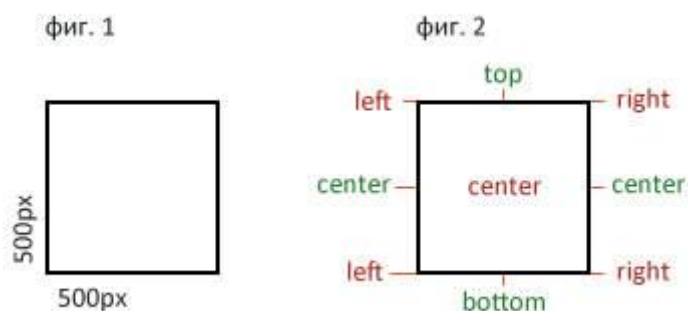
Пример: Отново имаме страница с размери 500 x 500px и снимка с размери 50px x 50px (фиг.1). Когато променим местоположението на изображението използвайки фиксирани стойности (например `background-position: 5px 10px;`), тогава изображението се премества 5px надясно (по хоризонтала x) и 10px надолу (по вертикала y) започвайки от горния ляв ъгъл.



Ако използваме само една фиксирана стойност, браузъра ще приеме като втора стойност ключовата дума "center".

Допустимо е използването на негативни стойности (например `background-position:-40px -15px;`).

- **keywords (ключови думи)** - Това са специални английски думи, които дават представа за място.



Хоризонтално позициониране:

- **left** - отговаря на лявата страна на елемента (съответства на 0%).
- **center** - отговаря на средната на елемента (съответства на 50%).
- **right** - отговаря на дясната страна на елемента (съответства на 100%).

Във всеки от горните случаи, ако дефинираме само хоризонталната позиция, вертикалната ще бъде 50% (което отговаря на **center**).

Вертикално позициониране:

- **top** - отговаря на горната страна на елемента (съответства на 0%).
- **center** - отговаря на средата на елемента (съответства на 50%).
- **bottom** - отговаря на долната страна на елемента (съответства на 100%).

Във всеки от горните случаи, ако дефинираме само вертикалната позиция, хоризонталната ще бъде 50% (което отговаря на **center**).

Следващата таблица ни представя как ключовите думи се отнасят към техните процентни еквиваленти:

1. `background-position: left top; /* като 0% 0% */`
2. `background-position: left center; /* като 0% 50% */`
3. `background-position: left bottom; /* като 0% 100% */`
- 4.
5. `background-position: right top; /* като 100% 0% */`
6. `background-position: right center; /* като 100% 50% */`
7. `background-position: right bottom; /* като 100% 100% */`
- 8.
9. `background-position: center top; /* като 50% 0% */`
10. `background-position: center center; /* като 50% 50% */`
11. `background-position: center bottom; /* като 50% 100% */`

- **inherit** - Тази стойност унаследява стойността приложена върху родителския елемент. Използва се много рядко.

Пример

Ще разгледаме само един пример, в който ще използвам фиксирани стойности в пиксели (px). Това обаче не трябва да те спира да експериментираш със стойностите в проценти или с ключовите думи.

Стъпка 1: **Ще използвам кода до който достигнахме в стъпка 5 от предишното свойство. HTML документа ще си остане същия. Промяната ще бъде само в CSS. Кода, който имаме до сега е следния:**

```

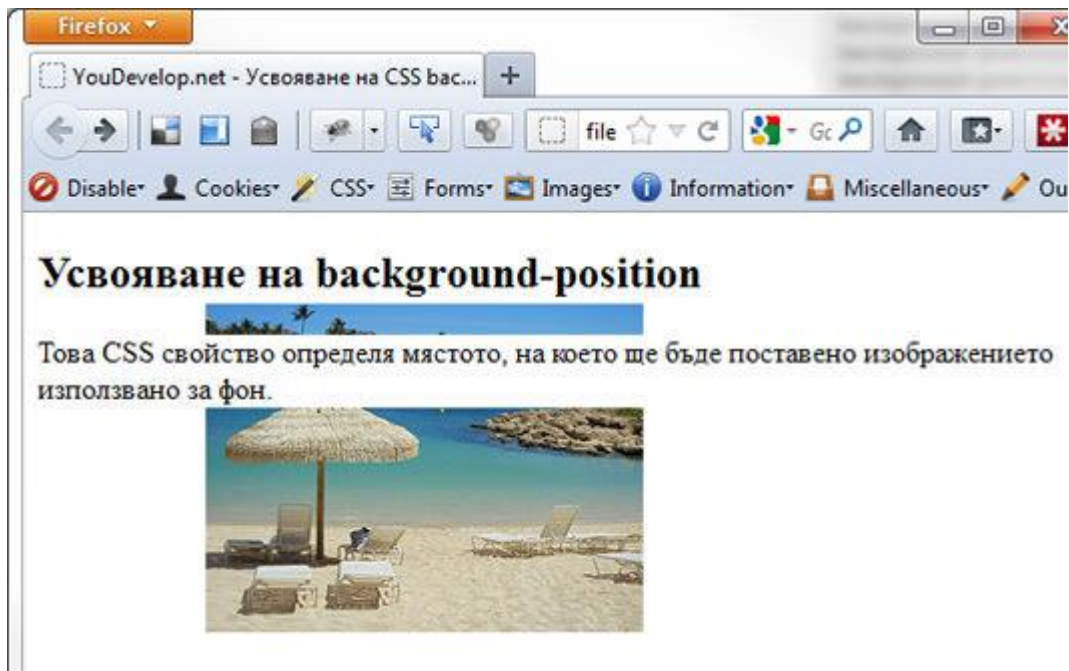
1.   body {
2.       background-image:url("snimka.jpg");
3.       background-repeat:no-repeat;
4.   }
5.   h2, p {
6.       background-color:white;
7.   }
```

Резултата от този код е фоново изображение, което не се повтаря, а се показва само веднъж в горния ляв ъгъл на страницата (виж резултата в стъпка 5 от примера на свойството [background-repeat](#)).

Стъпка 2: Нека предположим, че искаме да преместим изображението 100px надясно и 50px надолу. *background-position* е точното свойство за тази работа. Промени кода по следния начин:

```
1.     body {
2.         background-image:url("snimka.jpg");
3.         background-repeat:no-repeat;
4.         background-position:100px 50px;
5.     }
6.     /* код */
```

Запази промените и виж резултата в браузър.



Както можем да предположим, изображението се премести на желаните координати.

Разгледахме и свойството [background-position](#).

Това свойство се използва страшно много при работата със спрайтове (Sprites). Това е доста популярна техника, която ни позволява да комбинираме няколко изображения в едно, след което с помоща на *background* свойството можем да контролираме кое точно изображение да бъде видимо за посетителя. Повече за тази техника в някой от бъдещите уроци.

V. background-attachment

Това е последното свойство от "фамилията" *background*. Неговата работа е да определи дали изображението, което използваме за фон, ще се скролва (scroll) заедно с елемента или ще остане фиксирано на мястото си.

background-attachment може да приеме 3 стойности:

background-attachment: { scroll | fixed | inherit };

- **scroll** - Това е стойността по подразбиране. Тя позволява на изображението да скролва заедно с документа. С други думи, когато използваме скрола, за да се предвижваме нагоре и надолу в страницата, изображението също се движи заедно с документа.

- **fixed** - Тази стойност предотвратява изображението от това да се скролва заедно с документа. Местоположението му остава фиксирано.
- **inherit** - Тази стойност унаследява стойността приложена върху родителския елемент. Използва се много рядко.

Пример

Стъпка 1: **Ще използвам кода до който достигнахме в стъпка 2 от предишното свойство (background-position).**

За да тестваме това свойство, трябва да добавим достатъчно съдържание докато плъзгача, с който се предвижваме нагоре-надолу в страницата, стане видим. За целта нека копираме няколко пъти текста, с който вече разполагаме.

За сега няма да обръщаме внимание на html документа. Добави колкото искаш текстови параграфи и заглавия, но бъди сигурен/на, че страницата ти има скрол.

Стъпка 2: **Нека променим CSS кода:**

```

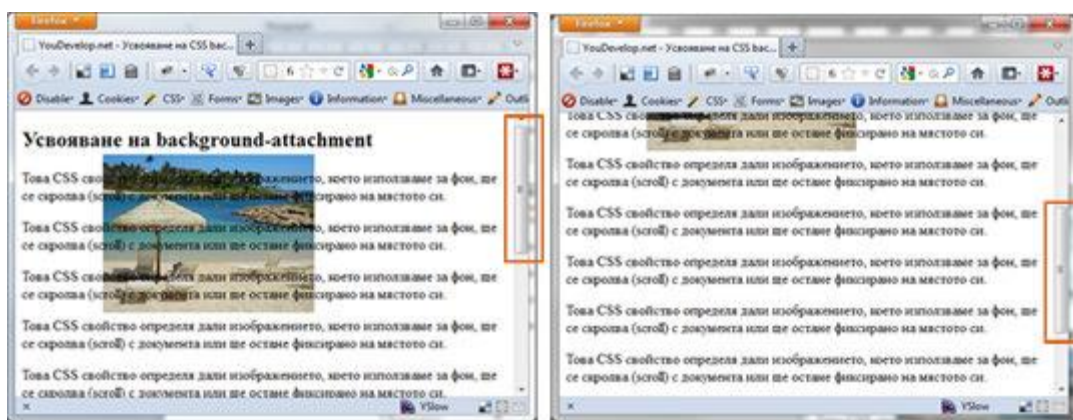
1.     body {
2.         background-image:url("snimka.jpg");
3.         background-repeat:no-repeat;
4.         background-position:100px 50px;
5.         background-attachment:scroll;
6.     }
7.     h2 {
8.         background-color:white;
9.     }

```

Единственото ново тук е, че премахнах белия фон от текстовите параграфи и също така към *body* добавих новото свойство *background-attachment* със стойност *scroll*.

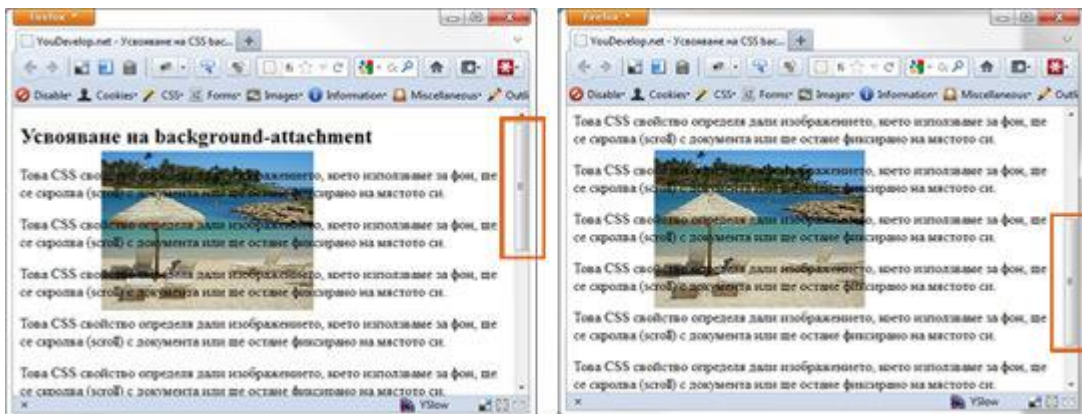
Всъщност, не е нужно отново да задаваме *background-attachment:scroll*, тъй като това е стойността по подразбиране. Все пак, за да бъде ясен примера ще го използвам.

В резултата, който ще получим виждаме, че изображението се скролва заедно с документа т.е. когато се движим нагоре и надолу в страницата, изображението също ще се "движи" заедно със скрола.



Стъпка 3: Ако променим *background-attachment* на *fixed*, изображението ще остане фиксирано на своето място и няма да се влияе от скрола на страницата.

1. /* код */
2. background-attachment:fixed;
3. /* код */



Разгледахме и последното *background* свойство. Надявам се, че всичко до тук е ясно и разбираемо. Ако имаш въпроси или коментари, сподели ги след урока.

VI. background (съкратено)

Това свойство е така нареченото **shorthand** или свойство, което обединява всички свойства от групата ("фамилията") на *background* свойствата.

До тук разгледахме свойствата [background-color](#), [background-image](#), [background-repeat](#), [background-position](#) и [background-attachment](#), като всяко заемаше нов ред и своя собствена декларация.

Общото свойство **background** ни позволява да **комбиниране всички свойства в едно**.

Пример

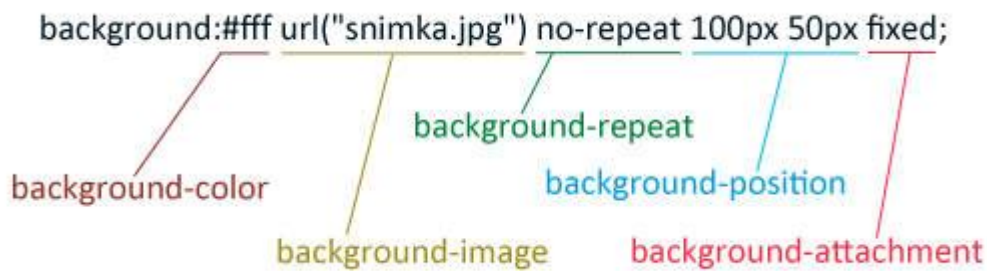
Ще използвам CSS кода от последния пример, който разгледахме. Към него ще добавя и *background-color*, за да изглежда по-ясно. Ето целия код, който прилагаме върху *body*.

1. background-color:#fff;
2. background-image:url("snimka.jpg");
3. background-repeat:no-repeat;
4. background-position:100px 50px;
5. background-attachment:fixed;

Посредством съкратеното **background** свойство, можем да сведем всичкия този код до 1 ред.

```
background:#fff url("snimka.jpg") no-repeat 100px 50px fixed;
```

Редът на задаване на свойствата е следния:



Не е проблем, ако пропуснем едно или повече свойства. В такъв случай, пропуснатите свойства приемат стойността си по подразбиране. Например:

```
background:url("snimka.jpg") no-repeat 100px 50px;
```

Тук съм пропуснал `background-color` и `background-attachment`. Те ще приемат съответно стойностите `transparent` и `scroll` (освен, ако не са презаписани в предишна декларация).

Това е краят на урока за усвояване на CSS свойството `background`. Постарах се да опиша всичко подробно и достатъчно разбираемо. Можеш да ползваш тези два урока като документ за справка, винаги когато имаш съмнения относно работата на `background`.

Ако мислиш, че урока ще бъде от полза на някой от твоите приятели, ще ти бъда благодарен, ако го споделиш с тях. Също така можеш да споделиш мнението си или да зададеш всякакви въпроси в коментарите след урока.

Стилизиране на текст чрез CSS

В този урок ще разгледаме следните CSS свойства:

- `line-height`
- `text-align`
- `text-decoration`
- `text-indent`
- `text-transform`
- `letter-spacing`
- `word-spacing`

1. `line-height`

Това е едно от най-важните и използвани свойства в уеб страницата като цяло, защото ни позволява да регулираме разстоянието между редовете.

Това означава, че можем да направим престоя на посетителите ни или ужасно труден, или ненаатрапчиво приятен.

Две от най-важните характеристики при работа с текст е той да бъде с приемливо голям и лесен за четене шрифт, и с достатъчно разстояние между редовете, което да позволява на текста да "диша".

Виж следните примери: (Текста е част от произведението на Елин Пелин - "Тераците")

Пример 1

Най-заможният човек в селото беше дядо Йордан Геракът. Пъргав и трудолюбив, той бе работил през целия си живот и бе сполучил да удвои и утрои имотите, останали от баща му. Надарен с ум практичен и с търговски способности, той бе съумял да направи и пари и да се издигне между селяните си като пръв човек.

Пример 2

Най-заможният човек в селото беше дядо Йордан Геракът. Пъргав и трудолюбив, той бе работил през целия си живот и бе сполучил да удвои и утрои имотите, останали от баща му. Надарен с ум практичен и с търговски способности, той бе съумял да направи и пари и да се издигне между селяните си като пръв човек.

Пример 3

Най-заможният човек в селото беше дядо Йордан Геракът. Пъргав и трудолюбив, той бе работил през целия си живот и бе сполучил да удвои и утрои имотите, останали от баща му. Надарен с ум практичен и с търговски способности, той бе съумял да направи и пари и да се издигне между селяните си като пръв човек.

В *Пример 1* текста е прекалено малък. Освен това разстоянието между редовете е също много малко, което прави параграфа труден за четене.

В *Пример 2* нещата изглеждат малко по-добре. Големината на шрифта е същата, но разстоянието между редовете е увеличено, което позволява на текста да “диша”. Параграфът е лесен за четене, но текста напраща очите заради малкия му размер.

Пример 3 ни предоставя оптимален вариант, при който текста е достатъчно голям и разстоянието между редовете е балансирано. Посетителя на нашата страница, без излишно натоваване върху очите, може изцяло да се фокусира върху четенето на текста.

Това е и целта на нашата страница, нали?

Да предоставим на посетителите ни добро изживяване и полезна, лесно откриваема информация, която не се крие в малки, трудни за четене редове.

Нека се върнем към **line-height**.

Това свойство може да приеме няколко стойности. Те са:

line-height: { length | number | percentage | normal | inherit };

- **length (дължина)** – Може да приеме различни величини. Формата на тази стойност е число (с или без десетична запетая) незабавно последвано от мерната единица (px, em, cm и др.). Когато стойността е 0, добавянето на мерна единица е по желание. Като цяло стойностите се делят на два вида: **relative (относителни)** и **absolute (абсолютни)**. И при двата вида негативните стойности са нелегални.

- **relative (относителни)** – Този тип стойности се определят на базата на други стойности за дължина. Те също се делят на два вида:

- **em** – Като база се взема стойността на *font-size* за съответния шрифт. Използват се често и могат да се адаптират към различни изходящи среди (например смартфон, таблет, компютър/лаптоп и т.н.);

- **ex** – Като база се взема стойността на *x-height* за съответния шрифт. Използва се много рядко;

Пример: h1 { line-height: 1.2em; }

Разстоянието между редовете на h1 заглавието ще бъде 1.2 пъти по-голямо от размера на шрифта (*font-size*) за този елемент (h1). Ако размера на шрифта е 24px, то *line-height* ще бъде $24 \times 1.2 = 28.8\text{px}$

- **absolute (абсолютни)** – Този тип стойности са фиксирани. Те най-често се използват, когато знаем изходящата среда. Тези мерни единици се състоят от физичните мерни единици и пиксела (px). Това са: **in, cm, mm, pt, pc** и **px**.

Пример: h1 { line-height: 22px }

Без значение от средата, в която ще отворим нашия сайт, разстоянието между редовете ще бъде 22px.

- **number (число)** – Тази стойност е число (с или без десетична запетая), което се умножава със стойността на *font-size* на елемента. Негативните стойности са нелегални.

Пример:

```
2. h1 {
3.   font-size:14px;
4.   line-height:1.5;
5. }
```

Резултата за *line-height* е $14\text{px} \times 1.5 = 21\text{px}$.

- **percentage (процент)** – Работата на тази стойност е идентична на предишната. Числото, което използваме се умножава със стойността на *font-size* на елемента. Негативните стойности са нелегални.

Пример:

```
6. h1 {
7.   font-size:14px;
8.   line-height:120%;
9. }
```

Резултата за *line-height* е $14\text{px} \times 120\% = 16.8\text{px}$.

- **normal** – Това е нормалното разстояние между редовете, което браузъра поставя по подразбиране. Обикновено това са стойности между 1.0 и 1.2.

- **inherit** - Унаследява стойността приложена върху родителския елемент.

В Стъпка 5 от [упражнението към урока Работа с параграфи](#) се запознахме с една лесна за запомняне и още по-лесна за прилагане формула, която ни помага да определим приблизително разстоянието между редовете.

Формулата е следната:

размер на шрифта x 1.5 = разстояние между редовете

Използвай я във всеки нов сайт, който създаваш, и в по-голямата част от случаите разстоянието между редовете ще бъде достатъчно.

Упражнение

Стъпка 1: Създай нов **html** документ и добави съдържание по твой избор, но бъди сигурен/а, че страницата съдържа поне 1-2 текстови параграфа. Също така създай и прикачи нов **CSS** файл, който ще използваме за стила на нашата страница. (За текст съм използвал първите два абзаца от разказа на Елин Пелин – “Гераците”). За да не удължавам излишно страницата, ще използвам няколко думи от началото и края на всеки абзац. В работните файлове към урока обаче, текста ще бъде в пълния си вид, **html** документа изглеждат ето така:

```
1. <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
2. "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
3. <html xmlns="http://www.w3.org/1999/xhtml">
4.   <head>
5.     <meta http-equiv="Content-Type" content="text/html;
6.       charset=utf-8" />
```

```

7.    <title>YouDevelop.net - Стилизиране на текст чрез CSS</title>
8.    <link href="style.css" rel="stylesheet" type="text/css" />
9.    </head>
10.   <body>
11.     <h1>Гераците - Първа глава</h1>
12.     <h4>Автор: Елин Пелин</h4>
13.     <p>Най-заможният човек в селото ...
14.     ... Затова всички го обичаха и почитаха.</p>
15.     <p>Неговата голяма и бяла къща, в която ...
16.     ... последните си дни много старци от рода им.</p>
17.   </body>
18. </html>

```

CSS документа съдържа следното:

```

1. body {
2.   font-family:Verdana;
3.   margin:0px auto;
4.   width:550px;
5. }
6. p {
7.   font-size:14px;
8. }

```

Страницата е центрирана в браузъра и има ширина от 550px. Шрифта е *Verdana*, което го прави лесен за четене. Размера на текста в параграфите е 14px.

Като за начало резултата не изглежда зле, но разбира се целта ни е да го подобрим.

Гераците - Първа глава

Автор: Елин Пелин

Най-заможният човек в селото беше дядо Йордан Геракът. Пъргав и трудолюбив, той бе работил през целия си живот и бе сполучил да удвои и утрои имотите, останали от баща му. Надарен с ум практичен и с търговски способности, той бе съумял да направи и пари и да се издигне между селяните си като пръв човек. Той имаше меко и добро сърце. И макар да беше малко скъперник, не беше строг в сметките си, помагаше на хората и се грижеше за селските работи. Затова всички го обичаха и почитаха.

Неговата голяма и бяла къща, в която живееше многолюдното му семейство, беше на лично място сред селото, виждаше се отдалече и отдалече личеше, че е чорбаджийска. Всред широкия двор, в който можеше да се смести една махала и който околоръст бе ограден като кале с бели зидове, се издигаше високо и право като стрела самотен кичест бор - единственият бор в цялата околия. Той бе донесен като малко борче от светите рилски гори и посаден тук в незапомнени времена от прадедите на Гераците. Това бе тяхното семейство знаме, с което се гордееха. Под неговата света сянка бяха отраснали няколко поколения. Под нея бяха преживели последните си дни много старци от рода им.

Стъпка 2: Нека увеличим разстоянието между редовете. Разгледай отново възможните стойности на свойството line-height и си избери няколко, които искаш да упражниш. В това упражнение аз ще работя с em стойност. Промени CSS кода по следния начин:

1. ...

```
2. p {
3.  font-size:14px;
4.  line-height:1.5em;
5. }
```

Разстоянието между редовете за всички параграфи ще бъде 1.5 пъти по-голямо от размера на текста: $14\text{px} \times 1.5 = 21\text{px}$.

Ако искаш да получиш същия резултат, но чрез проценти, тогава трябва да въведеш 150%.

Препоръчвам ти да отделиш няколко минути и да експериментираш с други стойности. Това е най-добрия начин да усвоиш всичко до тук.

2. text-align

Това свойство определя начина, по който съдържанието ще се подравни. Може да бъде приложено само върху блоков елемент. Може да приеме няколко стойности:

text-align: { center | justify | left | right | inherit };

- **center** – Текста ще се центрира базирайки се на ширината на родителския елемент;
- **justify** – Текста ще се подравни от двете страни (ляво и дясно). В някои случаи се наблюдава прекомерно разстояние между думите.
- **left** – Текста ще се подравни от ляво;
- **right** – Текста ще се подравни от дясно;
- **inherit** - Унаследява стойността приложена върху родителския елемент.

Когато системата на четене е от ляво на дясно, по подразбиране текста ще бъде подравнен от ляво, освен ако не се приложат допълнителни настройки.

Когато системата на четене е от дясно на ляво, по подразбиране текста ще бъде подравнен от дясно, освен ако не се приложат допълнителни настройки.

Упражнение

Нека използваме кода от упражнението към свойството [line-height](#). Единственото, което ще допълним е CSS кода. Тъй като нашата система на четене е от ляво на дясно, браузъра по подразбиране ще подравни текста от ляво, затова нека упражним някои от другите стойности.

```
1. ...
2. p {
3.  font-size:14px;
4.  line-height:1.5em;
5.  text-align:justify;
6. }
```

Резултата е следния (откъс от страницата):

Автор: Елин Пелин

`text-align:justify;`

Най-заможният човек в селото беше дядо Йордан Геракът. Пъргав и трудолюбив, той бе работил през целия си живот и бе сполучил да удвои и утрои имотите, останали от баща му. Надарен с ум практичен и с търговски способности, той бе съумял да направи и пари и да се издигне между селяните си като пръв човек. Той имаше меко и добро сърце. И макар да беше малко скъперник, не беше строг в сметките си, помагаше на хората и се грижеше за селските работи. Затова всички го обичаха и почитаха.

Автор: Елин Пелин

`text-align:center;`

Най-заможният човек в селото беше дядо Йордан Геракът. Пъргав и трудолюбив, той бе работил през целия си живот и бе сполучил да удвои и утрои имотите, останали от баща му. Надарен с ум практичен и с търговски способности, той бе съумял да направи и пари и да се издигне между селяните си като пръв човек. Той имаше меко и добро сърце. И макар да беше малко скъперник, не беше строг в сметките си, помагаше на хората и се грижеше за селските работи. Затова всички го обичаха и почитаха.

Автор: Елин Пелин

`text-align:right;`

Най-заможният човек в селото беше дядо Йордан Геракът. Пъргав и трудолюбив, той бе работил през целия си живот и бе сполучил да удвои и утрои имотите, останали от баща му. Надарен с ум практичен и с търговски способности, той бе съумял да направи и пари и да се издигне между селяните си като пръв човек. Той имаше меко и добро сърце. И макар да беше малко скъперник, не беше строг в сметките си, помагаше на хората и се грижеше за селските работи. Затова всички го обичаха и почитаха.

3. text-decoration

Това свойство определя декорацията, която ще бъде приложена върху текстовото съдържание на даден елемент. Може да приеме няколко стойности:

`text-decoration: { { blink | line-through | overline | underline } ... | none | inherit };`

- **blink** – Кара текста да премига. Голяма част от браузърите не поддържат тази стойност;
- **line-through** – Прокарва хоризонтална черта през текста;
- **overline** – Прокарва хоризонтална черта над текста;
- **underline** – Прокарва хоризонтална черта под текста;
- **none** – Не поставя декорация върху текста;
- **inherit** - Унаследява стойността приложена върху родителския елемент;

Упражнение

Ще използвам същите файлове, върху които работихме до сега. Нека приложим това свойство върху h4 заглавието. След промяната CSS кода ще придобие следния вид:

```
1. ...
2. h4 {
3.   color:gray;
4.   text-decoration:underline;
5. }
```

Запази промените и виж резултата в браузър.

```
text-decoration:underline;
```

Автор: Елин Пелин

```
text-decoration:line-through;
```

~~Автор: Елин Пелин~~

```
text-decoration:overline;
```

Автор: Елин Пелин

```
text-decoration:none;
```

Автор: Елин Пелин

4. text-indent

Ако си спомняш в училище госпожата по Български език все ни повтаряше, че когато започнем с писането на нов абзац е добре първият ред да бъде *“два пръста по-навътре”*. Това свойство ни позволява да направим точно това. След като го приложим, първият ред започва по-навътре (или по-навън, ако използваме негативна стойност).

Това свойство може да приеме няколко стойности:

```
text-indent: { length | percentage | inherit };
```

- **length (дължина)** – Може да приеме същите стойности като **length** стойността на свойството [line-height](#). Всичко, което важи за тази декларация, важи и тук. Негативните стойности са разрешени;
- **percentage (процент)** – Процент от ширината на елемента;
- **inherit** - Унаследява стойността приложена върху родителския елемент; Стойността по подразбиране е 0.

Упражнение

Ще продължа с подобряването на кода, с който разполагаме понастоящем. Ще направим една малка промяна в стила, който прилагаме върху текстовите параграфи.

```
1. ...  
2. p {  
3.   font-size:14px;  
4.   line-height:1.5em;  
5.   text-align:justify;  
6.   text-indent:1.5em;  
7. }
```

Резултата ще изглежда по следния начин.

Автор: Елин Пелин `text-indent:1.5em;`

Най-заможният човек в селото беше дядо Йордан Геракът. Пъргав и трудолюбив, той бе работил през целия си живот и бе сполучил да удвои и утрои имотите, останали от баща му. Надарен с ум практичен и с търговски способности, той бе съумял да направи и пари и да се издигне между селяните си като пръв човек. Той имаше меко и добро сърце. И макар да беше малко скъперник, не беше строг в сметките си, помагаше на хората и се грижеше за селските работи. Затова всички го обичаха и почитаха.

Автор: Елин Пелин `text-indent:10%;`

Най-заможният човек в селото беше дядо Йордан Геракът. Пъргав и трудолюбив, той бе работил през целия си живот и бе сполучил да удвои и утрои имотите, останали от баща му. Надарен с ум практичен и с търговски способности, той бе съумял да направи и пари и да се издигне между селяните си като пръв човек. Той имаше меко и добро сърце. И макар да беше малко скъперник, не беше строг в сметките си, помагаше на хората и се грижеше за селските работи. Затова всички го обичаха и почитаха.

5. text-transform

Това свойство контролира как и дали текстовото съдържанието на определен елемент ще бъде изписано с главни букви.

Може да приеме няколко стойности:

`text-transform: { capitalize | lowercase | none | uppercase | inherit };`

- **capitalize** – Превръща първата буква от всяка дума в главна. Всички други букви остават непроменени;
- **lowercase** – Превръща всички букви в малки;
- **none** – Текста приема нормалния си вид;
- **uppercase** – Превръща всички букви в главни;
- **inherit** - Унаследява стойността приложена върху родителския елемент;

Упражнение

Нека приложим това свойство върху h1 заглавието. HTML кода ще си остане непроменен. В CSS документа, след кода който имаме до сега, въведи следното:

1. ...
2. h1 {
3. text-transform:uppercase;
4. }

Всички букви от заглавието ще бъдат превърнати в главни. Нека видим резултата:

text-transform:uppercase;

ГЕРАЦИТЕ - ПЪРВА ГЛАВА

text-transform:capitalize;

Гераците - Първа Глава

text-transform:lowercase;

гераците - първа глава

text-transform:none;

Гераците - Първа глава

6. letter-spacing

Това свойство ни позволява да добавим допълнително разстояние между символите (букви, цифри и др.) в текстовото съдържание на определен елемент.

Може да приеме няколко стойности:

letter-spacing: { length | normal | inherit };

- **length (дължина)** – Тук важат същите правила като при свойството [line-height](#). Негативните стойности са разрешени;
- **normal** – Стойността по подразбиране;
- **inherit** – Унаследява стойността приложена върху родителския елемент;

Упражнение

Нека подобрим h1 заглавието още малко. Стила му ще придобие следния вид:

1. ...
2. h1 {
3. text-transform:uppercase;
4. letter-spacing:2px;
5. }

Резултат:

letter-spacing:2px;

ГЕРАЦИТЕ - ПЪРВА ГЛАВА

letter-spacing:-2px;

ГЕРАЦИТЕ - ПЪРВА ГЛАВА

letter-spacing:normal;

ГЕРАЦИТЕ - ПЪРВА ГЛАВА

7. word-spacing

Това свойство е идентично с [letter-spacing](#). Разликата тук е, че то ни позволява да добавим допълнително разстояние между думите в текстовото съдържание на определен елемент.

Може да приеме няколко стойности:

letter-spacing: { length | normal | inherit };

- **length (дължина)** – Тук важат същите правила като при свойството [line-height](#). Негативните стойности са разрешени;
- **normal** – Стойността по подразбиране;
- **inherit** – Унаследява стойността приложена върху родителския елемент;

Упражнение

В това упражнение ще променим разстоянието между думите в h4 заглавието. CSS кода ще придобие следния вид:

1. h4 {
2. color:gray;
3. text-decoration:none;
4. word-spacing:10px;
5. }

Резултат:

word-spacing:normal;

Автор: Елин Пелин

word-spacing:10px;

Автор: Елин Пелин

word-spacing:-5px;

Автор:ЕлинПелин

Това е края на този урок. Разгледахме няколко полезни CSS свойства, които ни дават контрол върху стила на текста на нашата уеб страница.

Надявам се, че урока беше ясен и разбираем. Ако имаш въпроси или искаш да допълниш нещо, можеш да го направиш чрез формата за коментари след урока.

Стилизиране на линкове

Чрез прилагането на стил върху нашата уеб страница, ние всъщност я открояваме от всички останали страници в интернет. Дизайн, функционалност, съдържание, стил и всичко останало са фактори, които я правят различна. Създаването на подходящ стил е също толкова важно, колкото и добавянето на оригинално съдържание. В този урок ще разгледаме няколко CSS и HTML техники, чрез които ще положим основите за подобряване на стила на линковете в нашата уеб страница.

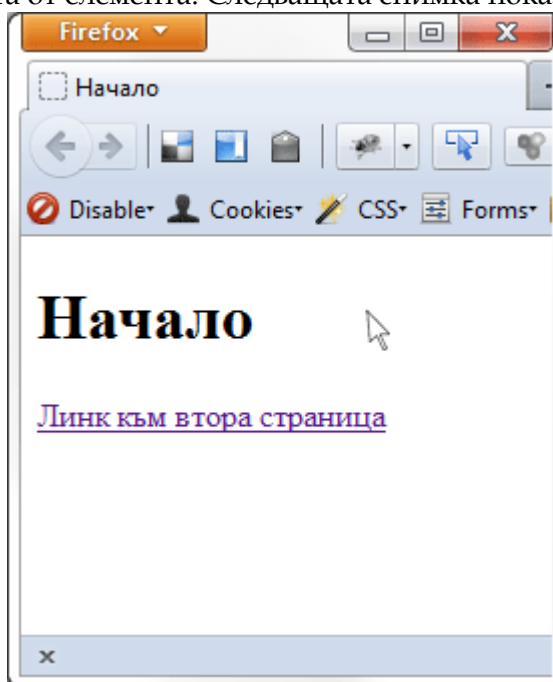
Структура на урока:

- [Добавяне на HTML атрибут title](#)
 - [Структура](#)
 - [Упражнение](#)

- [Промяна на цвят, фон, размер, шрифт, рамка и декорация на линк](#)
 - [Упражнение](#)
- [Промяна на стил на линк при mouse hover](#)
 - [Упражнение](#)
- [Увеличаване на активното пространство на линк](#)
 - [Упражнение](#)

1. Добавяне на HTML атрибут title

Атрибута **title** е един често използван атрибут, който служи за предоставяне на допълнителна информация към (почти) всеки елемент от уеб страницата. Обикновено, съдържанието на този атрибут се представя като “балонче”, което се показва при поставяне на курсора на мишката върху конкретния елемент. Балончето е видимо до момента, в който премахнем мишката от елемента. Следващата снимка показва работата на **title**.



Най-често, title атрибута се прилага към линкове (елемента `<a>`) като целта му е да предостави допълнителна информация за дестинацията на линка.

Title не е задължителен атрибут и използването му може да варира от линк до линк. Понякога самият текст на линка е достатъчно ясен и няма нужда от допълнително разяснение. Друг път показването на “балонче” може да разсея посетителя и да понижи доброто му преживяване в страницата (например при прилагане на title към линковете в главното меню). Като цяло ситуацията може да определи дали е необходимо добавянето на допълнителна информация или не.

Title атрибута се поставя в отварящия таг на желанния елемент. Стойността му може да бъде само текст. Следващия пример показва използването на title в елемента `<a>`:

```
<a href="" title="">Текст на линка</a>
```

И все пак кога е добре да се използва title? Като правило запомни, че title има за цел да подобри преживяването в страницата като предостави допълнителна информация за елемента. В никакъв случай title **не е заместител** на оригиналното и качествено съдържание. Когато имаш съмнение, че посетителя може би ще има нужда от допълнителна информация, за завършването на определена задача, тогава е точното време да му я предоставиш.

Пример:

- Когато от текста на линка не е напълно ясно къде ще го отведе. Добра идея е в този момент да обясниш с няколко думи дестинацията на линка;
 - Когато текста е достатъчно ясен, но кликането върху него ще предизвика отварянето на нов прозорец. Това също е удобен момент да предупредим посетителя за тази евентуална промяна като добавим текст *“Линка ще се отвори в нов прозорец”*.
 - Когато искаме да окуражим посетителя да извърши някакво действие. Например, на страницата ни има иконка на принтер, а при поставяне на мишката върху нея може да добавим текст *“Принтирай тази страница”*.
- Това са само няколко примера, но ситуациите могат да бъдат безкрайно много.

Упражнение

Ще създадем една обикновена html страница, която ще съдържа изображение и текстов линк. Линка ще води към Flickr страницата на автора заснел изображението. Към двата елемента ще приложим кратко описание с атрибута title.

Стъпка 1: Създай нов html документ и въведи следния код:

```
1. <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
2. "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
3. <html xmlns="http://www.w3.org/1999/xhtml">
4.   <head>
5.     <meta http-equiv="Content-Type" content="text/html;
6.       charset=utf-8" />
7.     <title>Атрибут title</title>
8.   </head>
9.   <body>
10.     <h1>Упражнение на title</h1>
11.     <p></p>
13.     <p><a href="http://www.flickr.com/photos/xjrlokix/2529980495/"
14.       title="Линк към снимката във Flickr">Благодарности на Ben
Fredericson</a></p>
15.   </body>
16. </html>
```

Стъпка 2: Страницата съдържа h1 заглавие последвано от два параграфа.

В първият параграф поставяме изображение, което се намира в една папка с html файла, затова в src атрибута въвеждаме единствено името на изображението последвано от файловото му разширение. В alt атрибута пък има кратък текст, който ще се покаже в случай, че снимката не успее да зареди. Title атрибута от своя страна съдържа текст, който ще бъде видим само когато посетителя сложи мишката върху снимката.

Вторият параграф съдържа текстов линк. В href атрибута въвеждаме адреса на страницата на фотографа във Flickr, а текста в title ни дава по-ясно описание на линка и евентуалното място, в което ще ни отведе.

Резултат:

Упражнение на title



[Благодарности на Ben Fredericson](#)

Ако намираш затруднение в упражнението, не се колебай да зададеш своите въпроси в коментарите след урока.

2. Промяна на цвят, фон, размер, шрифт, рамка и декорация на линк

От урока [“Прилагане на CSS чрез външен файл”](#) научихме, че чрез CSS можем да контролираме стила на всеки елемент съдържащ се в <body> на нашата страница. В това число, разбира се, се включват и всички линкове.

По подразбиране браузъра прилага свой собствен стил върху всеки линк от нашата страница.

Всеки линк от своя страна има 4 ключови състояния, които трябва да се имат в предвид при прилагането на стил. Всички те могат да се контролират със CSS чрез т.нар. псевдо-класове (pseudo-classes). Тези състояния са:

- **:link** – Съответства на линковете, с които посетителя все още не е взаимодействал т.е. не е посетил страницата, към която линка би го отвел;
- **:visited** – Обратно на предишното. Съответства на всички посетени линкове;
- **:hover** – Състояние, при което мишката е поставена върху линка (елемента);
- **:active** – Състояние, при което посетителя е кликнал върху линка, но е задържал бутона на мишката. Това състояние се активира в периода между кликването върху бутона на мишката и неговото освобождаване.

Визуално, стила на гореспоменатите състояния изглежда така:

[Линк](#) :link

[Линк](#) :visited

[Линк](#) :hover

[Линк](#) :active

Упражнение

Целта ни в следващото упражнение ще бъде да стилизираме нормалното състояние на линка като използваме псевдо-класа **:link**.

Стъпка 1: Ще се нуждаем от нов **html** документ, към който ще прикачим **CSS** файл. Кода в двата документа ще изглежда така:

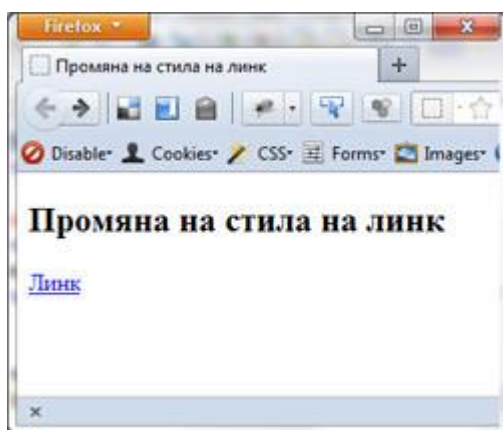
Index.html

```
1. <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
2. "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
3. <html xmlns="http://www.w3.org/1999/xhtml">
4.   <head>
5.     <meta http-equiv="Content-Type" content="text/html;
6.       charset=utf-8" />
7.     <title>Промяна на стила на линк</title>
8.     <link href="style.css" type="text/css"
9.       rel="stylesheet" />
10.    </head>
11.    <body>
12.      <h2>Промяна на стила на линк</h2>
13.      <p><a href="#">Линк</a></p>
14.    </body>
15.  </html>
```

style.css

```
1. p a:link {}
```

Резултата до тук е следния:



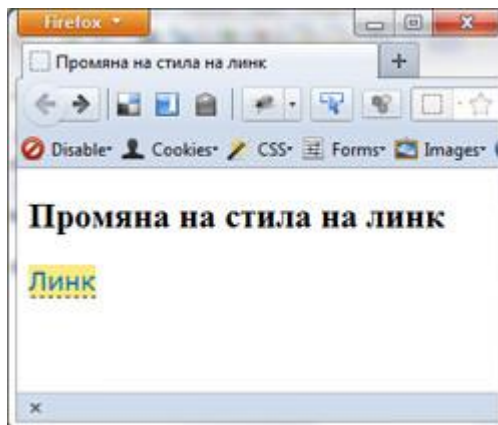
Стъпка 2: Селектора, който използваме е **p a**, което значи, че селектираме всички **<a>** елементи (линкове) намиращи се в елемента **<p>** (параграф). Чрез псевдо-класа **:link** пък прихващаме първото състояние на линка. Нека въведем малко **CSS**.

Отвори файла **style.css** и въведи следния код:

```
1. p a:link {
```

2. background-color:#ffea82;
3. border-bottom:1px dashed black;
4. color:#006eb7;
5. font-family:Verdana,Arial,sans-serif;
6. font-size:18px;
7. text-decoration:none;
8. }

Кода по-горе ще ни даде следния резултат:



Обяснение на декларациите:

- *background-color:#ffea82;* - определя фона на елемента. В урока [Усвояване на CSS background \(част 1\)](#) можеш да научиш повече за това свойство
- *border-bottom:1px dashed black;* - поставя черна, прекъсната линия с 1px дебелина в долната страна на елемента
- *color:#006eb7;* - променя цвета на текста
- *font-family:Verdana,Arial,sans-serif* - контролира шрифта на текста
- *font-size:18px;* - променя размера на текста на 18px
- *text-decoration:none;* - Стойността *none* премахва декорация, която браузъра прилага по подразбиране върху всеки линк. Повече за това свойство можеш да научиш в урока [Стилизиране на текст чрез CSS](#)

Отгук насетне всеки следващ линк, който отговаря на избрания селектор, ще се приеме този стил. Пробвай го.

Интересно е да се отбележи, че останалите 3 състояния на линка също приеха стила поставен върху **a:link**. Единствената разлика е, че браузъра все още прилига върху текста цвета по подразбиране на посетителите страници.

По същия начин по-горе можем да стилизираме и останалите 3 състояния.

3. Промяна на стил на линк при mouse hover

Добра практика е по някакъв начин да променим стила на линка, в случаите когато поставим мишката върху него (т.нар. mouse hover). Това състояние е идеалната възможност да предоставим на посетителя мигновен отзвук на действията му, което пък има за цел да подобри престоя му в страницата.

За да прихванем това състояние, ще използваме друг псевдо-клас - **:hover**. Той ни предоставя възможност да приложим специален стил само и единствено в случаите, когато мишката попадне върху конкретния елемент.

Упражнение

В html документа добави втори линк като копираш параграфа още веднъж.

index.html

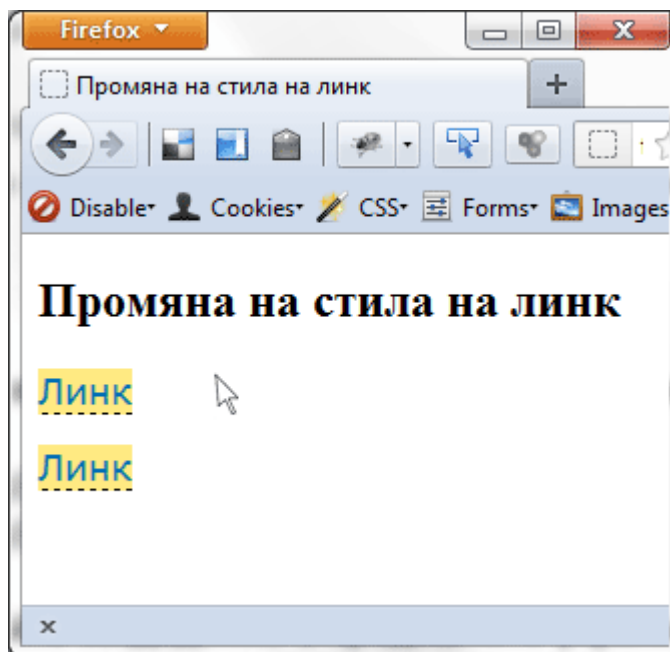
1. `<p>Линк</p>`
2. `<p>Линк</p>`

Сега отвори CSS файла, който създадохме в предишното упражнение, и добави следния код:

1. ...
2. `p a:hover {`
3. `background-color:#006eb7;`
4. `color:#ffea82;`
5. `}`

При mouse hover променяме фона и цвета на текста. Тези две декларации презаписват декралациите поставени по-рано чрез селектора **p a:link**. Всички други остават непроменени.

Резултат:



4. Увеличаване на активното пространство на линк

`<a>` елемента принадлежи към групата на инлайн елементите (в урока [Разлики между block и inline елементи](#) можеш да научиш повече за тях), което значи че ширината и височината му се определят от съдържанието.

За активно пространство на един линк се счита пространството върху и около линка, което го прави активен. Обикновено това се забелязва, когато поставим мишката върху линка (mouse hover) и тя се промени на ръчичка показваща ни, че можем да кликнем върху него.

Когато активното пространство е прекалено малко, това прави линка труден за активиране (т.е. кликане върху него), защото е по-трудно да бъде уцелен. Понякога е добра идея изкуствено да се увеличи активното пространство, което да направи линковете по-достъпни.

Това може да се постигне изключително лесно чрез използването на **CSS свойството padding**.

Упражнение

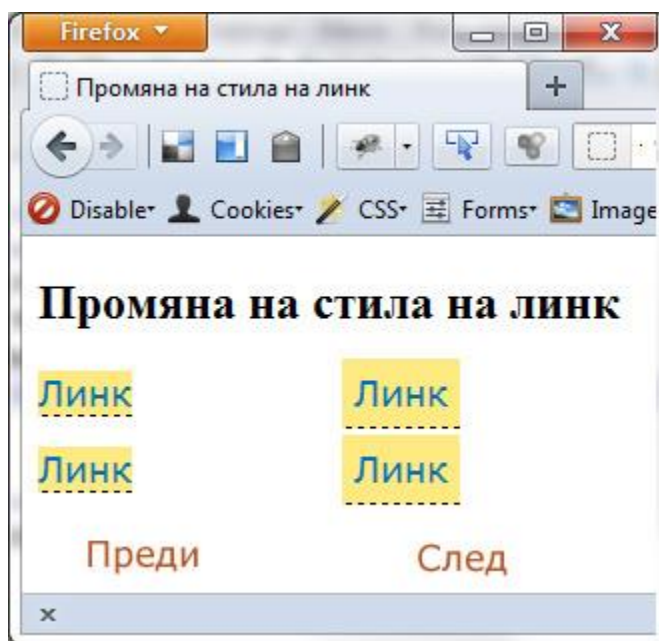
Отвори CSS файла style.css и в селектора **p a:link** добави следната декларация:

1. p a:link {
2. ...
3. padding:6px;
4. }

Стойността 6px прибавя допълнителни 6px към всяка от страните (горе, дясно, долу, ляво) на елемента <a>. Padding свойството ни позволява да увеличим/намалим вътрешното разстояние между ръба и съдържанието на избрания елемент.

Не се тревожи, ако в момента не разбираш всичко, padding свойството ще бъде тема на някой от бъдещите уроци.

В резултата виждаме много по-достъпни и лесни за уцелване линкове.



До тук разгледахме само няколко техники за стилизиране на линкове. Всъщност начините са безкрайно много, но общото между тях е, че са базирани върху използването на псевдо-класовете **:link**, **:visited**, **:hover** и **:active**.

Не се колебай да експериментираш в своята следваща уеб страница и се постарай да направиш стила ѝ уникален.

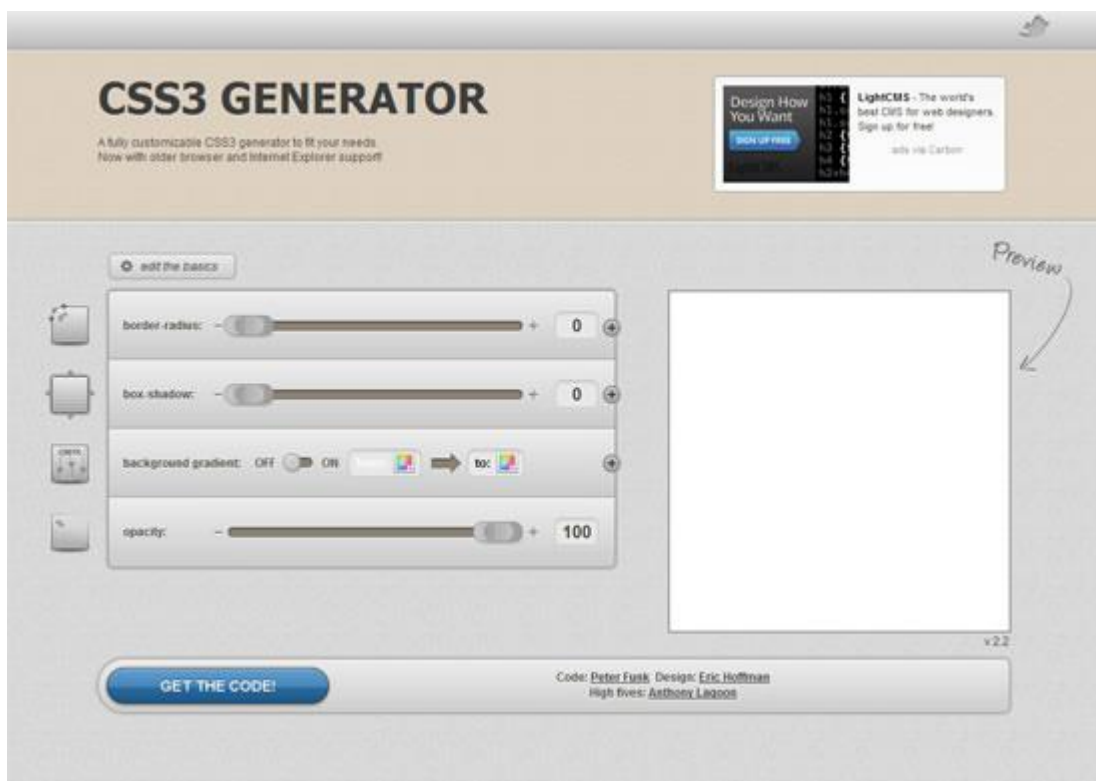
Ако имаш въпроси относно упражненията към урока или искаш да споделиш своя коментар, можеш да го направиш чрез формата след урока.

Навярно вече знаеш за разширените възможности, които CSS 3 ни предоставя. Благодарение на тях неща като оформяне на заоблени ъгли, прилагане на сянка към елемент и дори създаване на анимация са по-лесни от всякога. И това е само началото.

Установяването на CSS 3 като стандарт е все по-близо и голям брой уеб сайтове вече успешно го прилагат. Добрата новина е, че от днес **ти също можеш да го използваш в своята уеб страница**. Следващите няколко уеб сайта са няколко страхотни онлайн инструмента, които ще ти помогнат да положиш основите.

1. CSS3.me - CSS3 Generator

Чрез своя приятен и интуитивен интерфейс, CSS3.me ни предоставя възможност да манипулираме няколко CSS3 свойства като заоблени ъгли (border-radius), добавяне на сянка върху елемент (box-shadow), фон с два преливащи един към друг цвята (background-gradient) и прозрачност на елемент (opacity). Правоъгълника в дясно показва крайния резултат в реално време.



Линк към сайта: <http://www.css3.me/>

2. CSS3 Generator

Този инструмент има доста по-разширени възможности от предишния като ни дава възможност в реално време да пробваме някои от най-популярните в момента CSS3 свойства. Това включва border-radius, box-shadow, text-shadow, @font-face, transitions, transforms и други.



Линк към сайта: <http://css3generator.com/>

3. CSS3 Maker

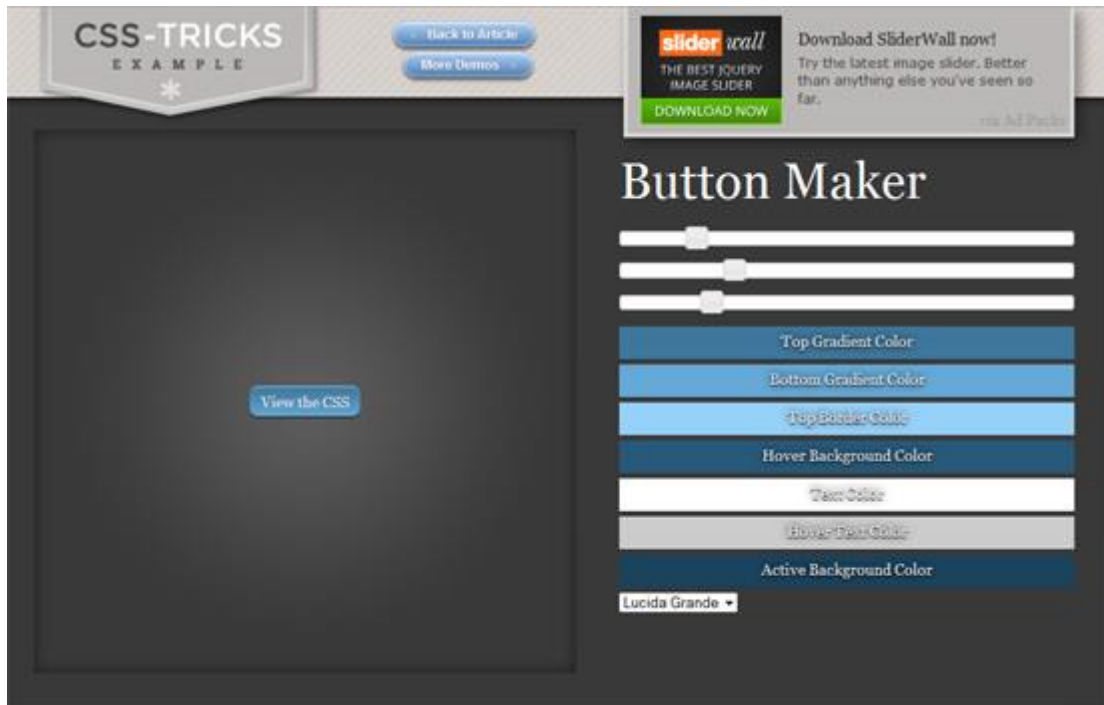
Един страхотен и мулти-функционален CSS3 инструмент, който заслужава внимание. Всяка страница ни дава възможност да пробваме начина на работа на съответното свойство като резултатите са видими в реално време. Освен това в правоъгълника "Browser Compatibility" виждаме поддръжката на съответното свойство в най-популярните браузъри.



Линк към сайта: <http://www.css3maker.com/>

4. CSS-Tricks Button Maker

Оригинален инструмент, който е създаден за едно единствено нещо и това е лесното изработване на бутони. Лявата част ни показва резултата в реално време базиран на параметрите в дясно. При клик върху бутона се отваря малък прозорец с кода на бутона.



Линк към сайта: <http://css-tricks.com/examples/ButtonMaker/#>

5. CSS3 Button Generator

Още един инструмент за създаване на бутони. В горната част на страницата виждаме крайния резултат. Под него се намират параметрите, които имаме възможност да променим, а в дясно от тях е HTML и CSS кода, който ще ни бъде нужен, за да поставим бутона в нашата уеб страница.



BACKGROUND	BORDER / PADDING	DROP SHADOW	INNER SHADOW
START COLOR: <input type="color" value="#ffff00"/>	COLOR: <input type="color" value="black"/>	COLOR: <input type="color" value="black"/>	COLOR: <input type="color" value="white"/>
<input type="checkbox"/> <input type="checkbox"/> 25	WIDTH: <input type="text" value="3"/> px	OPACITY: <input type="text" value="0.5"/>	OPACITY: <input type="text" value="0.7"/>
<input type="checkbox"/> <input type="checkbox"/> 50	HEIGHT: <input type="text" value="30"/> px	X: <input type="text" value="0"/> px	H: <input type="text" value="0"/> px
<input type="checkbox"/> <input type="checkbox"/> 75	TOP + BOTTOM: <input type="text" value="10"/> px	Y: <input type="text" value="1"/> px	V: <input type="text" value="0"/> px
END COLOR: <input type="color" value="#ffcc00"/>	LEFT + RIGHT: <input type="text" value="20"/> px	SIZE: <input type="text" value="3"/> px	SIZE: <input type="text" value="1"/> px

TEXT FONT	TEXT SHADOW 1	TEXT SHADOW 2
COLOR: <input type="color" value="black"/>	COLOR: <input type="color" value="black"/>	COLOR: <input type="color" value="white"/>
SIZE: <input type="text" value="14"/> px	OPACITY: <input type="text" value="0.4"/>	OPACITY: <input type="text" value="0.3"/>
	H: <input type="text" value="0"/> px	H: <input type="text" value="0"/> px
	V: <input type="text" value="-1"/> px	V: <input type="text" value="1"/> px
	SIZE: <input type="text" value="0"/> px	SIZE: <input type="text" value="0"/> px

SAVE + SHARE

[ツイート](#) 1,850
 [mixi Check](#)
 [Dribbble](#) 343
 [+1](#) 170

Like 600 people like this. Be the first of your friends.



HTML CODE

```
<button type="button" name="" value="" class="css3button">
```

CSS CODE

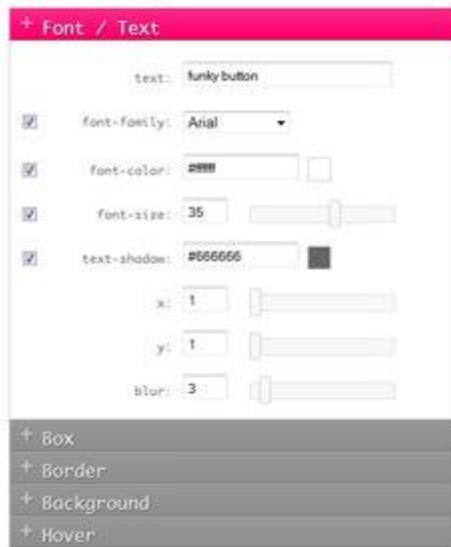
```
button.css3button {
font-family: Arial, Helvetica, sans-serif;
font-size: 14px;
color: #050505;
padding: 10px 20px;
background: -moz-linear-gradient(
top,
#ff7f00 0%,
#ff0000);
background: -webkit-gradient(
linear, left top, left bottom,
from(#ff7f00),
to(#ff0000));
border-radius: 30px;
-moz-border-radius: 30px;
-webkit-border-radius: 30px;
border: 3px solid #000000;
-moz-box-shadow:
0px 1px 3px rgba(0,0,0,0.5);
-webkit-box-shadow:
0px 1px 3px rgba(0,0,0,0.5);
text-shadow:
0px -1px 0px rgba(0,0,0,0.4),
0px 1px 0px rgba(255,255,255,0.3);
}
```

Линк към сайта: <http://css3button.net/>

6. CSS3 Button Generator

Ако предишните два инструмента ти се струват недостатъчни, ето още един. Параметрите са грижливо групирани в няколко раздела като промяната на всеки от тях се отразява директно върху бутона видим в дясно.

the settings



+ Font / Text

text: funky button

font-family: Arial

font-color: #000000

font-size: 35

text-shadow: #666666

x: 1

y: 1

blur: 3

+ Box

+ Border

+ Background

+ Hover

the button



Линк към сайта: <http://css3buttongenerator.com/>

7. CSS Gradient Generator

Този инструмент ни дава възможност да създадем преливащи един към друг цветове подобно на Photoshop. В горната лява част на страницата има голямо количество от готови за използване шаблони. Естествено, можем да създадем и наши собствени чрез използването на плъзгачите.

Ultimate CSS Gradient Generator

A powerful Photoshop-like CSS gradient editor from [ColorZilla](#).

For Firefox For Chrome Gradient Generator

Presets



Name: Blue Gloss Default

save



Stops

Opacity: <input type="text" value="100"/>	Location: <input type="text" value="0"/>	delete
Color: <input type="text" value="#000000"/>	Location: <input type="text" value="100"/>	delete

Adjustments

hue/saturation... reverse

Sponsor



Manage Projects Better
Get your projects done faster, easier with
Zoho Projects. 1 Project Free!

via Ad Fads

Preview



Orientation: vertical ↓

Size: 370 x 50

IE

CSS

switch to scss

```
background: rgb(30,87,153); /* Old browsers */
background: -ms-linear-gradient(top,
  rgba(30,87,153,1) 0%, rgba(41,137,216,1) 50%,
  rgba(32,124,202,1) 51%, rgba(125,185,232,1)
  100%); /* FF3.6+ */
background: -webkit-gradient(linear, left top,
  left bottom, color-
  stop(0%,rgba(30,87,153,1)), color-
  stop(50%,rgba(41,137,216,1)), color-
  stop(51%,rgba(32,124,202,1)), color-
  stop(100%,rgba(125,185,232,1))); /*
  Chrome, Safari4+ */
background: -webkit-linear-gradient(top,
  rgba(30,87,153,1) 0%, rgba(41,137,216,1)
  50%, rgba(32,124,202,1)
  51%, rgba(125,185,232,1) 100%); /*
  Chrome10+, Safari5.1+ */
background: -o-linear-gradient(top,
  rgba(30,87,153,1) 0%, rgba(41,137,216,1)
  50%, rgba(32,124,202,1)
  51%, rgba(125,185,232,1) 100%); /* Opera
  11.10+ */
background: -ms-linear-gradient(top,
  rgba(30,87,153,1) 0%, rgba(41,137,216,1)
  50%, rgba(32,124,202,1)
  51%, rgba(125,185,232,1)
  100%);
```

Color format: rgba

Comments

IE9 Support (?)

import from image

import from css

Permalink

Link to, save or share the current gradient using its unique link.

Линк към сайта: <http://www.colorzilla.com/gradient-editor/>