**препоръчителни ресурси:**

<http://www.wikihow.com/Design-a-Website>

<http://www.w3.org/2002/03/csslayout-howto><http://www.w3.org/TR/CSS2/visuren.html#absolute-positioning> <http://www.ironspider.ca/webdesign102/tablesvscss.htm> <http://reference.sitepoint.com/css/positioning>

X <http://blog.teamtreehouse.com/css-positioning>

**намерени. но не прегледани:**

[1] http://moodle.le.tsdoit.org/course/view.php?id=13, Система за електронно обучение за курса по World-Wide Web технологии, последно посетена на 2013-03-26 (основен-онлайн)

[2] Programming Windows 8 Apps with HTML, CSS, and JavaScriptURL Move right Move Update Delete Hide Assign roles, http://moodle.le.tsdoit.org/mod/url/view.php?id=456 (допълнителна-книга)

[3] http://moodle.openfmi.net/course/view.php?id=550 (Курс Уеб приложения - дистанционен)

<http://code.google.com/speed/page-speed/docs/rules_intro.html>

1. **Основни свойства при CSS позиционирането**

Когато изграждаме интерфейса на един софтуерен продукт е от изключителна важност всеки елемент да заема точно определно място. Понякога това е задача, която изисква познания как би се държал дадения елемент в различни ситуации. Спрямо конкретната му функция, можем да изпълним тази задача по различни начини. Сега ще се запознаете с някои от основните методи.

1. **Свойството position и различните му стойности**

По подразбиране, елементите в един html документ се разполагат в реда на тяхното деклариране. Поради различните нужди и желани крайни резултати, това последователно подреждане може да бъде манипулирано. Сега ще разгледаме четирите най-използвани стойности на css свойството **position:**

* 1. **static -** Статично позициониране на елемент

Първата стойност, която ще разгледаме е тази, която е “default”-ната за всеки елемент. Ето и пример, за това как се декларира тя:

.example-selector {

position: static;

}

В повечето случаи, употребата на стойността “static” е напълно излишна, защото тя се задава по подразбиране, ако няма друга декларирана стойност. По този начин, елементите в html-документа ще се подредят в последователен ред, в зависимост от това дали те са блокови (пр. <div></div>) или вградени/inline (пр. <span></span>).

* 1. **relative -** Релативно позициониране на елемент

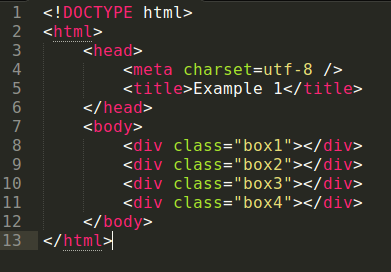
Другата стойност, която е изключително честа и удобна в прилагането на различни техники, е relative.

.example-selector {

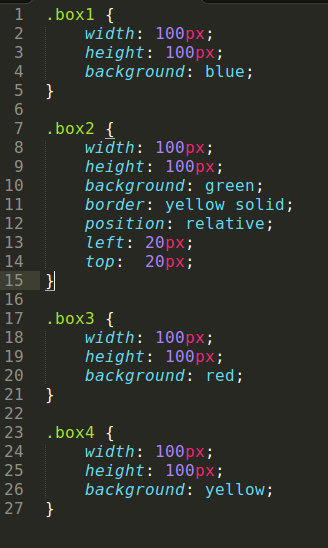
position: relative;

}

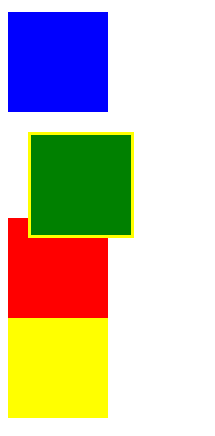
Сигурно се питате каква е разликата между тези две стойности? Тези два елемента биха се държали напълно еднакво, ако просто има зададем стойност static или relative, с изключение на това, че когато има релативно разположен обект, то на него могат да му се декларират други четири CSS свойства (top, right, bottom, left). По този начин да манимулираме първоначално зададената му позиция. За целта нека разгледаме примерния код [1] по-долу и след това да го анализираме.



*код 1. Деклариране на 4 блокови елемента*



*код 2. Деклариране на стиловете за дефинираните елементи в код 1*



*снимка 1 : резултат от код 1 и код 2*

Първо, тези кутии ще се покажат в реда, в който сме ги декларирали в **код 1,** като всяко ще бъде на нов ред, заради блоковия елемент <div></div>. Но както се вижда на изображението от крайния резултат, втората кутия е изместена от очакваното място. Това се дължи на дефинираните й две свойства е **код 2(** top: 20px; left: 20px; ).

По този начин, тази кутия е изместена отгоре и отляво с по 20px от първоначалното си място. Тук е важно да припомним, че координатната система, която се използва и спрямо която се разполагат и изчисляват позициите на елементите, е с център горе вляво (както още се нарича, Картезийска координатна система в 4-ти квадрант или още Декартова). Подобен ефект ще се получи, ако използваме и други две свойства (margin-top, margin-left), но тяхната употреба ще повлияе на останалите елементи. Когато задаваме позицията на някой елемент със стойности на top, right, bottom и left, не влияем по никакъв начин на разположението на околните елементи.

* 1. **absolute -** Абсолютно позициониране на елементи

Идва ред и на третата възможна стойност и примера за нейната декларация:

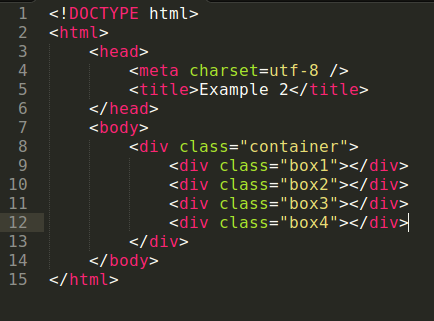
.example-selector {

position: absolute;

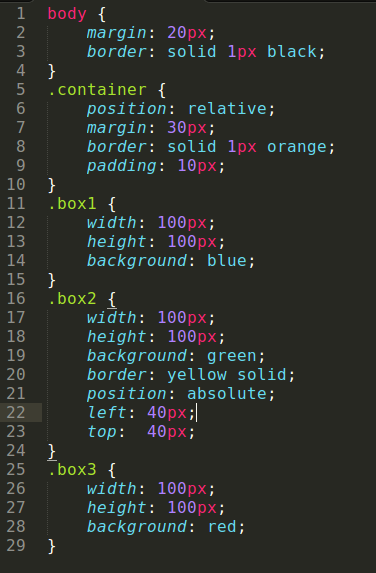
}

При позициониране на някой елемент със стойност absolute, то той е напълно отделен от нормалния поток на елементите и подобно на релативното разполагане на обекти, може да му се прилагат вертикални и хоризонтални отклонения с помощта на вече познатите top, right, bottom и left. Но разбира се, съществува някаква разлика между тези стойности. Както казахме, абсолютно позиционирания елемент е напълно отделен от последователността на елементите. Това означава, че околните елементи ще се държат така, сякаш той дори не е там.

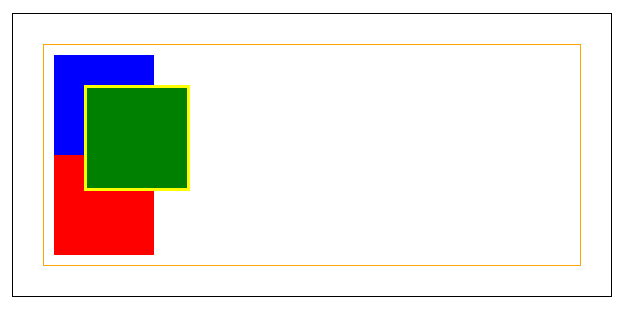
Когато прилагаме методи да отместим обект, с деклариран “position: relative”, ние приемаме неговата изходна точка мястото, където е неговото първоначално подреждане в потока на елементи. Ето тук идва и съществената разлика на тези две стойности (relative, absolute), защото при отместване на обект с абсолютна стойност, неговото отместване ще бъде спрямо прозореца на браузъра ( или документа, т.е. <html></html> ). Но това може да се промени като декларираме “position: relative” на елемента-контейнер, спрямо който искаме да отместим елемента. По този начин се създава нов контекст на позициониране. За да стане по-ясно това, нека разгледаме следния код и прилежащия резултат от него [2]:



*код 3: деклариране на 4 блокови елемента, разложени в общ блоков контейнер*

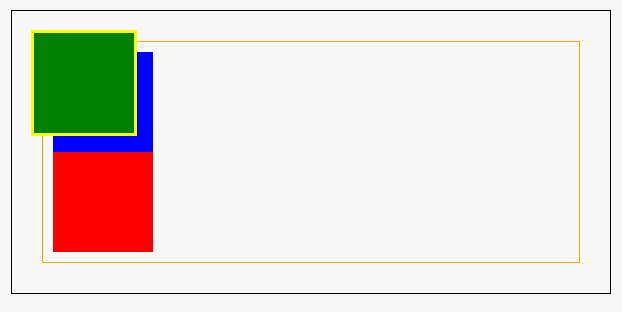


*код 4: деклариране на стиловете за дефинираните елементи в код 3*



*снимка 2: резултат от код 3 и код 4*

В този пример, оранжевия контейнер, в който са разположени четирите кутии, е дефиниран с “position: relative” като по този начин създава новото контекстово пространство. Със сега зададените стойности за отместване на зелената кутия (top: 40px; left: 40px;), тя се отмества спрямо този контейнер. Ако на този контейнер не му беше зададена релативна стойност или изобщо го нямаше, то зелената кутия щеше да се отмести със същите стойности, но спрямо прозореца на браузъра и би изглеждало така:



*снимка 3: резултат от код 3 и код 4, без с промяна на контекстовото пространство*

Важно е отново да се отбележи, че с този начин на отместване на елементите, не сме засегнали никой от останалите в документа.

* 1. **fixed -** Фиксирано позициониране на елементи

Ето и позледната стойност за свойството position.

.example-selector {

position: fixed;

}

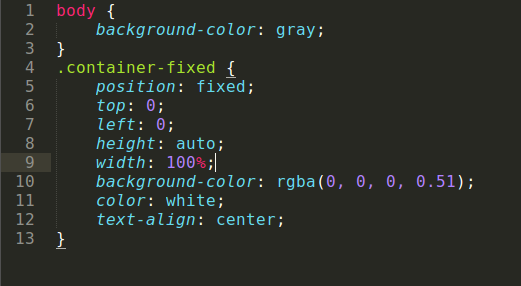
Спрямо спецификацията си, фиксирано разположените елементи попадат така да ссе каже в категорията на абсолютно позициониране. Тези два типа са много подобни, но с две ключови разлики:

* при фиксираните елементи контекстовото пространство е винаги началото на документаи и не се влияе от предшествани релативно дефинирани контейнери както при абсолютно зададени елементи.
* точно заради това фиксирано място, когато се scroll-ва страницата, този елемент няма да се мести заедно с останалото съдържание на документа.

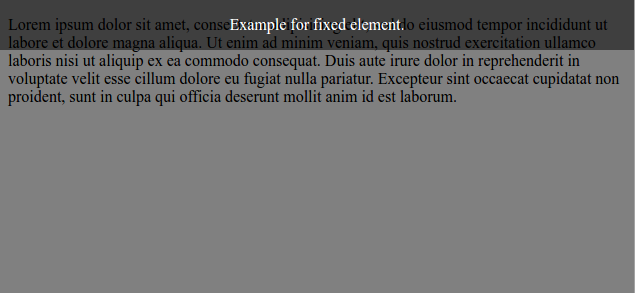
Нека разгледаме следния примерен код и резултат, за да се онагледи дефиницията:



*код 5: Деклариране на два контейнера - съдържателен текст и фиксиран контенер*



*код 6: стилове към код 5*



*снимка 4: резултат от код 5 и код 6*

В горната част на страницата се показва полупрозрачна лента с текст в нея и тя ще остане там дори и при scroll-ване на страницата. Това е постигнато като на този контейнер е деклариран с “position: fixed”. Тук отново можем да използваме свойството z-index и по този начин да оформим уникален стек (наслояване) от отделни елементи, в зависимост от нуждата ни.

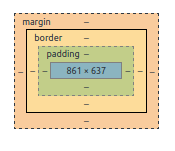
* 1. Съществуват и други три стойности, които могат да използват, но за тях все още няма добра поддръжка от браузърите. Това са стойностите center, page, sticky. За сега само ще ги споменем и ако желаете малко повече информация за тях, може да разгледата следните линкове - <https://www.w3.org/TR/css-position-3/#center-positioning>, <https://www.w3.org/TR/css-position-3/#page-positioning>, https://css-tricks.com/almanac/properties/p/position/#article-header-id-5

1. **Margin, padding**

Употребата на тези две свойства понякога е доста объркваща. Сега ще се опитам да обясня разликата между тези две свойста и как те засякат пространстово от елементи.

В общи линии, елементите в CSS са представени като правоъгълна кутия, чийто размер се определя от елементите content, padding, border, margin.

Content е съдържанието на елемента, padding заобикаля това съдържание, border-a го обгражда, а margin е вънщния слож на елементите. Ако погледнете следната снимка, ще ви стане малко по-ясно:

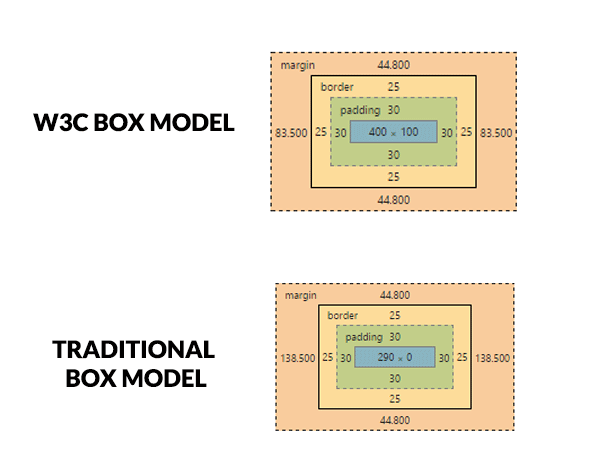


*снимка 5: Диаграма на отделните слоеве на един елемент.*

Както се вижда от диаграмата, padding-a е слой, който се простира от границата на елемента до неговото съдържание и се използва, за да се контрулира това разстояние. Приложената стойности на padding-a ще влияе само на този елемент. Ако желаем да постигнем някакъв контрол между разстоянието на отделните елементи в пространството без да засегнем самия размер на елемента, то за тази цел трябва да използваме свойството margin (или съответните негови подсвойства margin-top, margin-right, margin-bottom, margin-left)

При работа с кутии е много важно да се има предвид, че размера на всички кутии зависи от използвания модел. Съществуват два модела:

* **W3C boxed model -** този моделът счита, че ширината на елемента е равна на съдържанието на кутията (content), с изключение на нейните padding и border, които се добавят върху всички величини, зададени за елемента
* **traditional boxed model -** oт друга страна, традиционният модел на кутията счита, че ширината на елемента е равна на сумата от content-a, padding-a и border-a, приложена към кутията. Това дава по-предсказуеми резултати и е по-лесно да се работи с тях.



*снимка 6: Двата вида box model [3]*

По подразбиране всеки браузър използва модела на W3C. Това може изрично да се посочи като се зададе стойности на свойството box-sizing - content-box (за W3C модела) или border-box (за традиционния модел). По-интуитивния модел от двата, традиционния, е най-популярният подход сред уеб програмистите. Със следващия пример, може да зададетеоразмеряването на кутиите да следват традиционния модел.

html {  
 box-sizing: border-box;  
}  
  
\*, \*:before, \*:after {  
 box-sizing: inherit;  
}

**Настройване на margin и padding**

* Когато задаваме единична стойности на някой от двете свойства (пр. padding: 10px;), това означава, че тя ще бъде приложена за всички страни. Ето следните възможности:

padding: 10px; / за всички страни/

padding: 10px 20px; / отгоре и отдолу по 10px | отляво и отдясно по 20px/

padding: 10px 15px 20px; / отгоре - 10, отляво и отдясно - 15, отдолу - 20 /

padding 10px 15px 20px 25px / отгоре -10, отдясно - 15, отдолу - 20, отляво - 25 /

Конвенцията за задаване на стоности на margin и border е същата. Но за по-голямо улеснения, има и кратки свойства - padding-top, padding-right, padding-bottom, padding-left, margin-top, margin-right, margin-bottom, margin-left, border-top, border-right, border-bottom, border-left.

Съществуват обаче някои тънкости при използването на тези елементи. За да може да запазим еднакъв изглед на елементите при различни устройства, така че винаги да постигаме еднакъв реултат, то трябва да се съобразяваме с някои неща. Примерно, ако сме задали размер на някой елемент в проценти, то е добре да съобразим че и стойнстите на margin и padding е също добре да бъдат в тази мерна единица. По този начин, когато гледаме документа в различни резолюции, тези разтояния ще са презметнат и по еднакъв начин.

Съществуват много тънкости в използването на различните мерни единици за постигане на желания краен резултат, но това са тема на друг реферат.

Цитирана литература:

[1] , [2] [**Louis Lazaris**](http://blog.teamtreehouse.com/author/lazaris), “CSS Positioning: A Comprehensive Look” , 13.11.2012

[<http://blog.teamtreehouse.com/css-positioning>], последно посетен на 26.04.2017г.

[3] [*Dave Maxwell*](https://github.com/davemaxwell77)*,* [*Adrian Sandu*](https://www.sitepoint.com/author/asandu/)*, and* [*Panayiotis Velisarakos*](https://www.sitepoint.com/author/pvelisarakos/), [ <https://www.sitepoint.com/set-css-margins-padding-cool-layout-tricks/> ], последно посетен на 29.04.2017г.

Съдържание:

1. Основни свойства при CSS позиционирането
   1. position
   2. margin and position
   3. dislplay - <https://www.w3schools.com/cssref/pr_class_display.asp>
2. Свойството position и различните му стойности
   1. static
   2. relative
   3. absolute
   4. fixed
   5. нови стойности
3. Margin, padding
4. Интересни

[1] http://css-tricks.com/snippets/css/a-guide-to-flexbox/

[2] <https://scotch.io/tutorials/a-visual-guide-to-css3-flexbox-properties>

<http://bootsnipp.com/snippets/featured/fancy-sidebar-navigation>

Версия 2

За презентация:

1. начален - тема + име
2. static
3. relative ---- + top, right, bottom, left
4. absolute -----
5. fixed
6. други практики - margin, position, display: flex - теми на други реферати от колегите