Cascading Style Sheets (CSS) версии 1, 2 и 3

История
Цели на CSS
Свойства
Стилово форматиране и представяне
Кутиен модел
Позициониране
Примери

Основни характеристики на документ

Съдържание	• Информация, съдържаща текст, графика, аудио или видео
Структура	• Подялба и последователност на информационните части
Оформление	• Онагледяване на съдържанието и структурата на документ
Метаданни	• Описание семантиката на съдържанието

Style sheet – a set of rules that specify the presentation of a document. Style sheets are written by an Author, and interpreted by a User Agent, to present the document to the User.

https://www.w3.org/TR/css-2022/

Каскадни стилове (Cascading Style Sheets)

- Cascading Style Sheets (CSS) е език за стилово представяне на външния вид и форматиране на маркирано съдържание в HTML, XML и XML-базиран формат.
- CSS предоставя прост механизъм за добавяне на стилови набори към структурирани Уеб документи, чрез който авторите и читателите на документите могат да влият на начините на представяне на съдържанието.
- Представянето, управлявано чрез CSS, може да се извърши от браузер, печатащо устройство, синтезатор на реч, телевизия и други медии.

CSS – малко история

- CSS се базира на стиловия език Document Style Semantics and Specification Language (DSSSL), разработен за дефиниране на стилове за SGML през 80-те години на миналия век.
- За разлика от DSSSL обаче, CSS позволява добавянето на повече от един стилови документа, които да управляват представянето на съдържанието в браузер.
- Понастоящем CSS е представена чрез набор от спецификации на W3C, достъпни през страницата http://www.w3.org/Style/CSS/.
- CSS е спецификация на W3C

Версии на CSS

1. CSS 1 спецификацията бе завършена през 1996 година – слаба поддръжка от браузерите (освен Opera – още през 1997). Internet Explorer 5.0 за Macintosh през март 2000 г. - първи браузър с почти пълна поддръжка на CSS 1.
2. CSS 2 е препоръка на W3C от май 1998, но многото проблеми с поддръжката й от браузерите доведоха до преразглеждане на стандарта CSS 2 в спецификацията CSS 2.1, която бе най-популярна от 2007г. до 2015г.
3. CSS 3 стартира още през далечната 1998 г. и вече е завършена. Поддръжа се напълно от известните браузери (http://www.w3.org/Style/CSS/).
4. W3C започва изготвянето и на CSS 4 от 2009 г. Няма CSS ниво 4. Независимите модули могат да достигнат ниво 4 или повече, но езикът CSS вече няма нива. Поддържа се частично от известните Уеб браузери.

Тествайте степента на поддръжка на CSS версиите на браузера ви онлайн на адрес: https://css4-selectors.com/browser-selector-test/

css4-selectors.com/browser-selector-test/

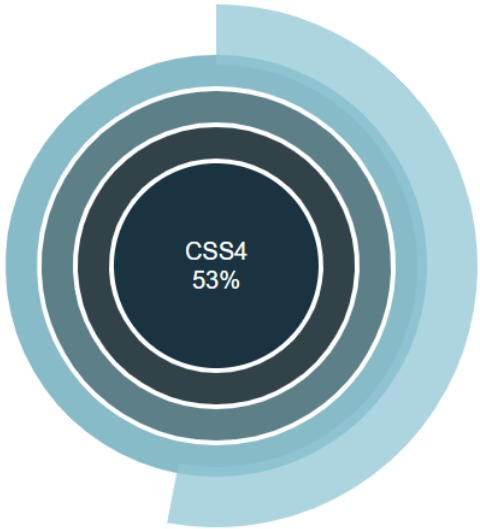
{CSS4} Selectors

HOME / SELECTORS / STATUS / BROWSER TEST

Browser CSS Selector Test

This browser selector test will check how well your browser supports the CSS selectors from level 1 up to the upcoming level 4.
After you started the test-suit which is based on the working draft, it will automatically check which selectors are supported by your browser, not necessarily if your browser implemented it correctly according to the specification. Please note that the specification isn't finalized yet, so the CSS selector (level 4) support is increasing at the moment, the browser fabricators are constantly working on implementing new or changed parts of the specification.

Browser test results



ноември 2019г.

css4-selectors.com/browser-selector-test/

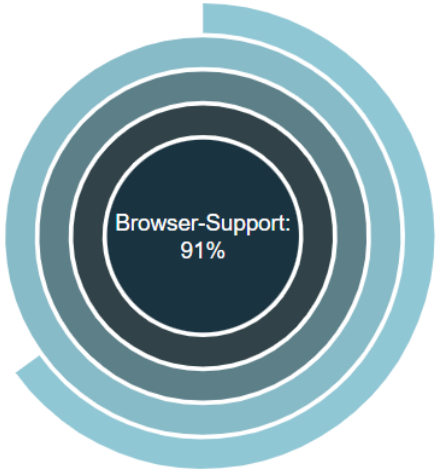
{CSS4} Selectors

HOME / SELECTORS / STATUS / BROWSER TEST

Browser CSS Selector Test

This browser selector test will check how well your browser supports the CSS selectors from level 1 up to the upcoming level 4.
After you started the test-suit which is based on the working draft, it will automatically check which selectors are supported by your browser, not necessarily if your browser implemented it correctly according to the specification. Please note that the specification isn't finalized yet, so the CSS selector (level 4) support is increasing at the moment, the browser fabricators are constantly working on implementing new or changed parts of the specification.

Browser test results



ноември 2022г.

Предимства на CSS 1/4

- Стиловото оформление допълва структурираното маркъп съдържание.
- Разделението между двете дава възможност на няколко страници да споделят едно форматиране и така да се намали сложността и повторенията в структурното съдържание (например да се работи с уеб дизайн без таблично структуриране).
- CSS предоставя по-голяма гъвкавост и контрол върху характеристиките на представяне и внасянето на промени.
- Чрез посочване на стилове за Уеб документите, дизайнерите могат да опростят поддръжка на Уеб страниците, като обаче запазят консистентен изглед и усещане за целия сайт.

Предимства на CSS 2/4

- CSS може да позволи една и съща страница от съдържанието да се представя в различни стилове за различни методи за показване в разнообразни медии, като например на екрана, в печатна форма, речеви синтезатор и Брайлова азбука.
- Една уеб страница да се показва по различен начин в зависимост от параметрите на крайното устройство, като размер на екрана, поддръжка на цветове и др.
- Авторът на документа може да зададе един стилов набор, всеки читател може да използва различен стил на представяне, а накрая клиентското устройство (напр. браузер) обикновено има свой вграден стил.

Предимства на CSS 3/4

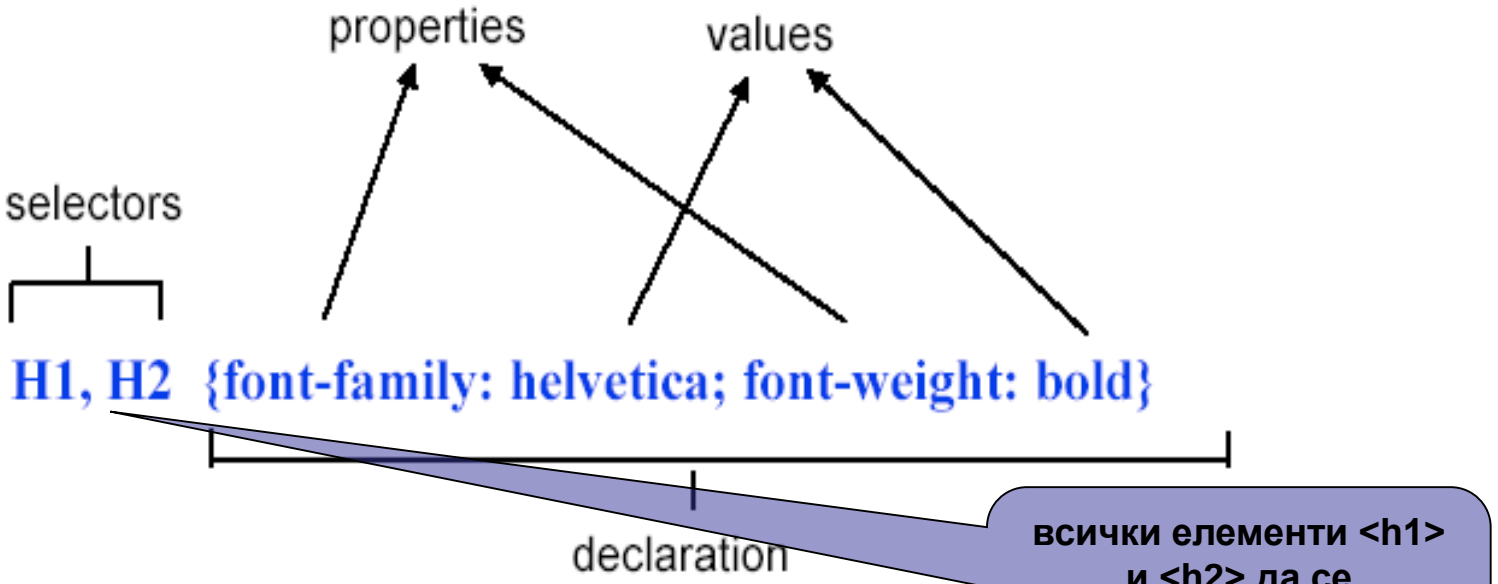
- CSS подобрява достъпността на съдържанието – хора с увреждания имат достъп до съдържанието през подходящи за тях медии и по създадено специално за тях представяне. Потребители със специфични изисквания към представянето могат да пренебрегнат зададения от автора CSS2 стил.
- CSS спецификациите са проектирани така, че да имат съвместимост отдолу нагоре.
  - CSS 2 потребителските клиенти са в състояние да интерпретират CSS1 стилови набори.
  - Същевременно, CSS 1 клиентите могат да четат CSS 2 стилове, като игнорират онези части от тях, които им са непознати.
  - Също така, потребителски агенти, които нямат поддръжка на CSS, могат да представят цялото съдържание без дефинираните CSS стилове.

Предимства на CSS 4/4

- CSS декларациите имат прост синтаксис и се съхраняват в текстови файлове, които могат лесно да бъдат манипулирани от външни приложения - едно приложение може да управлява ефективно начина на представяне в дадена медия на съдържание в HTML, XML и XML-базиран формат, единствено чрез манипулиране на стойностите в CSS декларациите.
- CSS е отворен стандарт, който осигурява независимост от платформи и производители на софтуер.
- CSS е лесно четим от потребителите и позволява бързо оформяне на стиловите набори. Той изразява стила в общата терминология на предпечатната подготовка

Представяне на CSS правила

- CSS представя стил на документа чрез комбинация от правила за представяне на определени елементи от документа (но не и на атрибути), които да се явят в медията (*за CSS 1 - единствено браузер*) по желан от дизайнера начин.
- Примерно CSS правило:



Структура на CSS правила

Правилото е разделено на две части:

- 1. списък от селектори - указва елемент или елементи, за които се прилага декларацията
- 2. декларация - определя как трябва да бъдат оформени елементите от списъка от селектори

CSS контекстов селектор

- Вместо разделен със запетая списък от няколко елемента се допускат и по-сложни модели, за да се показват йерархии от елементи. Така например декларацията

P EM { background: yellow }

задава контекстов селектор (указва, че подчертаният текст в рамките на параграф трябва да има жълт фон; подчертаният текст на други места няма да бъде засегнат).

- Контекстовият селектор представлява низ от два или повече прости селектори, разделени от празно пространство. На такива селектори могат да се зададат стилови свойства и поради правилата на каскадния ред *те ще имат предимство пред простите селектори*.

Селектори за клас

- Простите селектори могат да имат различни ***класове***, с цел използване на различни стилове за даден елемент. Например за показването на съдържанието на code в различен цвят можем да използваме:
  - **code.html { color: #191970 }**
  - **code.css { color: #4b0082 }**
- Така създаваме два класа - **css** и **html** за ползване за елемента **CODE** в HTML's. Атрибутът **CLASS** се ползва в HTML за означаване на класа на елемента, *напр. за*
  - **P.warning { color: #191970 }**
  - **ползваме**
  - **<P CLASS=warning>Only one class is allowed per selector. For example, code.html.proprietary is invalid.</p>**

В HTML може и без кавички.

Още за класовете

- Класове може да се декларират и без асоциран елемент:
  - **.note { font-size: small }**
  - В този случай, класът **note** може да се използва с всеки елемент!
- Добра практика е да назовем класовете в съответствие с тяхната функция, а не с външния им вид. Класът **note** в горния пример би могъл да се нарича **small**, но това име ще стане безсмислено, ако авторът е решил да промени размера на шрифта.

Стилове в HTML

- За да свържете външен списък със стилове към документ, добавете елемента **LINK** в хедъра на вашия документ:
  - **<LINK> – an EMPTY element**

- **Атрибути за елемента LINK:**
  - **id** ID #IMPLIED -- SGML ID attribute --
  - **href** CDATA #IMPLIED -- URL for linked resource -
  - **rel** CDATA #IMPLIED -- *defines the relationship between the linked file and the HTML document. REL=StyleSheet specifies a persistent or preferred style*
  - **title** CDATA #IMPLIED -- advisory title string --
  - **type** CDATA #IMPLIED -- advisory Internet media type --

Използване на LINK за свързване с външна CSS дефиниция

```
<LINK TITLE="Stanford" REL="stylesheet"
      HREF="http://www.stanford.edu/stanford.dsssl"
      TYPE="application/dsssl">

<LINK TITLE="Cal" REL="stylesheet"
      HREF="http://www.berkeley.edu/cal.css"
TYPE="text/css">
<h1>Welcome to my amazing home page!</h1>

   If your browser supports style sheets, try this page
in Cal and Stanford styles!!
```

Атрибут MEDIA (от CSS2)

Елементът **<LINK>** приема опционален атрибут **MEDIA**, който специфицира медията, за която се прилага стила. Възможни стойности са:

- **screen** (стойност по подразбиране), за презентиране на екран;
- **print**, за изход на принтер;
- **projection**, за презентиране на прожектори;
- **aural**, за синтезатори на реч;
- **braille**, за Брайлови у-ва;
- **tty**, за телетипни у-ва (с фиксиран шрифт);
- **tv**, за телевия;
- **all**, за всякакви у-ва.

*Многократни типове медия се задават в списък разделени със ‘,’ или с **all**.*

Вътрешни стилове – чрез елемента STYLE

- Използван за вграждане на стилове в документа, чрез елемента **STYLE**:

```
<STYLE TYPE="text/css" MEDIA=screen>
<!--
    BODY { background: url(foo.gif) red; color: black }
    P EM { background: yellow; color: black }
    .note { margin-left: 5em; margin-right: 5em }
-->
</STYLE>
```

- Елементът **STYLE** се разполага в **HEAD**.
- Изискваният атрибут **TYPE** определя CSS тип.

Импортиране на външни стилове в CSS файл

- Стилове могат да се импортират с клаузата **@import** – може да се използва както в CSS файла, така и в **STYLE** елемент:
- <STYLE TYPE="text/css" MEDIA="screen, projection">
  <!--
  @import url(http://www.htmlhelp.com/style.css);
  @import url(/stylesheets/punk.css);
  DT { background: yellow; color: black }
  -->
  </STYLE>
- Забележете, че:
  - Други CSS правила могат да бъдат включени в **STYLE** елемент, но всички **@import** клаузи трябва да се зададат в началото на CSS документа.
  - Всички правила в импортиращия документ предефинират тези от импортираните стилове.

Вграден (Inlining) стил

- Начин да приложим стил към единичен таг е да използваме атрибута **STYLE**. **STYLE** може да се приложи към всеки елемент на **BODY** с изключение на **BASEFONT**, **PARAM**, и **SCRIPT**. Стойността на атрибута е една или повече двойки свойство /стойност разделени с двоеточие. Например:
- **<P STYLE="color: red; font-family: 'New Century Schoolbook', serif">** This paragraph is styled in red with the New Century Schoolbook font, if available.</P>
- Забележете, че **New Century Schoolbook** е ограден с ‘ в атрибута **STYLE** (а не с “).
- Inlining стиловете не се препоръчват – липса на гъвкавост. *Защо?*

Атрибути за поддръжка на стилове

Повечето от тях са разрешени в много елементи:

- **ID** – за рефериране към стил в даден елемент
- **CLASS** – разрешава набор от елементи да се групират по стил, зададен от класа
- **STYLE** - съдържа стилова информация за представяне на елемента

ID селектори

ID селекторите индивидуално се определят за целите на представяне на даден елемент. Те трябва да се използват пестеливо, поради своите присъщи ограничения. ID селектор се задава чрез използване на индикатор "#", предхождащ името му. Например :

- **#svp94O { text-indent: 3em }**
  ще се реферира в HTML документа по **ID** атрибут:
- **<P ID=svp94O>Text indented 3em</P>**

*An **em space** is the width and height of the capital letter M in the current font size and design.*

CLASS и STYLE атрибути

- **CLASS**:
  - **<P CLASS ="special">**
    Стиловото правило може да зададе всички елементи от стила **CLASS** "**special**" например да бъдат с определен цвят.

- **STYLE**:
  - **<P STYLE ="*... style sheet right here !...*">**
    - Зададеният по-горе ***style sheet*** ще замени стила по подразбиране, описан чрез HTML **META** елемент.

HTML елементи, подпомагащи стилизирането

- **SPAN -** разграничава (маркира) за целите на стилизирането *част от текста, която не притежава структурно значение*. Може да има атрибути:
  id, class, style
- **DIV** - разграничава (маркира) за целите на стилизирането *поредица от елементи*.

SPAN и DIV примери

- **<DIV CLASS="section">**
  - **<P>The spec states that DIV, together with the CLASS attribute, is used to denote structural elements, like "abstract" or "chapter", for which HTML doesn’t provide elements.**
  - **<P>DIV allows headers, lists, paragraphs, and other DIV elements.**
  - **<P> <SPAN CLASS="init">Maybe we want</SPAN> the first three words here to be set in a special way. SPAN elements can appear wherever you expect text.**
  - **<P>Presumably, somewhere, a style sheet would have to state how the "init" and "section" classes are to appear. Or, instead, we could have used the STYLE attribute and put a style statement in directly in with the tags.**
- **</DIV>**

Свойства, стойности и групиране

- **Свойства (Properties)**

  *Свойството (**property**) се задава на селектор за манипулиране на стила му*. Примери за свойства: **color**, **margin** и **font**.

- **Стойности (Values)**

  Стойността (***value***) се присвоява в декларацията на дадено свойство (***property***). Например, свойството **color** може да получи стойността **red**.

- **Групиране**

  Групирането намалява повторенията на дефинициите. Например за всички оглавления (headers) можем да зададем:

H1, H2, H3, H4, H5, H6 { color: red; font-family: sans-serif }

Видове CSS свойства 1/2

- текст и шрифт – в тази група влизат свойствата, които задават какъв да бъде шрифта (**font-style**), неговата големина (**font-size**), дебелина (**font-weight**), разстоянията между буквите (**letter-spacing**) и други.
- цвят и фон – включва свойства за цвят на текст или очертание (**color**), цвят на фон (**background-color**), изображение за фон (**background-image**) и др.
- кутиен модел – съдържа свойства, свързани с кутийно представяне на текст като дебелина на очертанията на кутийното представяне (**border-width**), неговия стил **(border-style**), цвят (**border-color**), височина (**height**), широчина (**width**) и др.

Видове CSS свойства 2/2

- позициониране – състои се от свойствата за позициониране като например разстоянието на стартовата позиция от най-горния десен ъгъл (**top**), от най-левия долен ъгъл (**bottom**), отдясно (**right**), отляво (**left**) и др.
- списъци – включва свойства за представяне на списък като например:
  - стил (**list-style**)
  - тип (**list-style-type**)
  - изображение за булет (**list-style-image**) и
  - позициониране (**list-style-position**).

Видове CSS елементни селектори 1/3

- универсален – означават се с знака * се прилагат към всички елементи. Пример: *** { }**
- типов – прилага се или поединично, или за повече елементи (разделени със запетая). Пример:

firstName, lastName, country { margin-top:100px; position:absolute;}

- пряк наследник – прилага се за елементи, които са преки наследници на други. Пример:
  parent_element > child_element {....}
- наследник – прилага се за всички елементи, които са наследници на даден елемент. За разлика от горния вид селектор, наследниците може да не са преки наследници. Пример: **element_parent element_descedant {....}**

Видове CSS елементни селектори 2/3

- **брат/сестра един след друг** – пример: за XML документа

```
<parent>
        <brother>…</brother>
        <sister>  </sister>
</parent>
```

- CSS правилото:

```
brother + sister{    margin-top:100px;
                     position:absolute;
}
```

- ще се приложи за елемента **sister**, ако той веднага следва след **brother.**

The ***adjacent sibling combinator*** (+) separates two selectors and matches the second element only:

1) if it immediately follows the first element, and
2) both are children of the same parent element.

Видове CSS елементни селектори 3/3

- **брат/сестра, където и да са** – пример: за XML документа

```
<parent>
        <brother>…</brother>
        ….
        <sister>  </sister>
</parent>
```

- CSS правилото:

```
brother ~ sister{    margin-top:100px;
                     position:absolute;
}
```

ще се приложи за елемента **sister**, ако той следва където и да е след **brother.**

the difference is that the second selector does **NOT** have to immediately follow the first one means It will select all elements that is preceded by the former selector.

Видове CSS елементни селектори - обобщение

- Универсален: *** { }**
- Типов: **element_1, element_2, … , element_n {....}**
- Пряк наследник: **parent_element > child_element {....}**
- Наследник: **element_parent element_descedant {....}**
- Брат/сестра (където и да са):

  brother ~ sister { … }
- Съседни брат/сестра:

  brother + sister { … }
- Клас за елемент:

  my_element.my_class{ … }

CSS селектори за атрибути

- Задаване:
  a[target] { background-color: yellow; }
  – селектира всички елементи <a> с атрибут target
- Атрибут с дадена стойност:
  a[target="_blank"] { background-color: yellow; }
  - избира <a> елементите с атрибут target="_blank"
- Атрибут, съдържащ дадена дума:
  [title~="flower"] { border: 5px solid yellow; }
  - всички елементи с атрибут title, съдържащ flower
- Атрибут, започващ с дадена (цяла!) дума – разл. от **[class^="top"]**:
  [class|="top"] { background: yellow; }
  - всички елементи с атрибут class, започващ с top
- Атрибут, завършващ с дадена стойност:
  [class$="test"] { background: yellow; }

Наследяване (Inheritance)

- Много от CSS свойствата дефинирани за даден елемент могат да бъдат наследявани от неговите под-елементи.
- Така веднъж дефинирани за даден елемент, те са в сила и за неговите под-елементи.
- Ако искаме обаче да заменим свойствата за даден под-елемент, то това може да стане чрез създаване на специфично свойство конкретно за него.

Наследяване - пример

За CSS дефиницията

```
h1, h2, h3 { color:#000000;
  margin-top:100px;
  position:absolute;
  font-weight:bold; }
```

- искаме да добавим свойство **font-style** само за **h2**:

```
h2 {font-style:italic;}
```

- или да предефинираме свойството **color** за тези елементи **h2**, които следват веднага след **h1:**

```
h1 + h2 {color:#000FFF;}
```

Псевдо- класове и елементи

Отнасят се за случаи, наподобяващи структурни елементи, без всъщност да представляват такива.

- *Котва (Anchor)* ***псевдо-класове*** – представят статуса на връзка
- *Типографски* ***псевдо-елементи*** – представят се като елемент, който се показва по време на изпълнение

Котва (Anchor) псевдо-класове

- Отнасят се до елемента **A** (котва) и се използват за *различно* показване на връзките (links), посетените връзки (visited links) и активните връзки (active links), въответно с имена на псевдо-клас:
- **link**
- **visited**
- **active**.
- Пример: по-голям шрифт и син цвят за активни линкове и по-малък шрифт и зелен цвят за посетени такива:

```
A:link { color: red }
A:active { color: blue; font-size: 125% }
A:visited { color: green; font-size: 85% }
```

First Line & Letter псевдо-елементи

- CSS1 включва възможността за различно представяне на първата линия от дадено съдържание чрез **first-line** псевдо-елемент – може да се ползва при всеки ***block-level елемент като напр.*** **P**, **H1**, и т.н., напр.:
- **P:first-line { font-variant: small-caps; font-weight: bold }**
- Псевдо-елементът **first-letter** се използва за така наречените бити букви (drop caps) и други ефекти. Първата буква от текста ще се представи със зададения стил. **first-letter** може да се ползва при всеки ***block-level елемент***. Пример:
- **P:first-letter { font-size: 300%; float: left }** указва бита буква, три пъти по-голяма от текста.

Изравняване с останалия текст (left | right | none).

Други примери

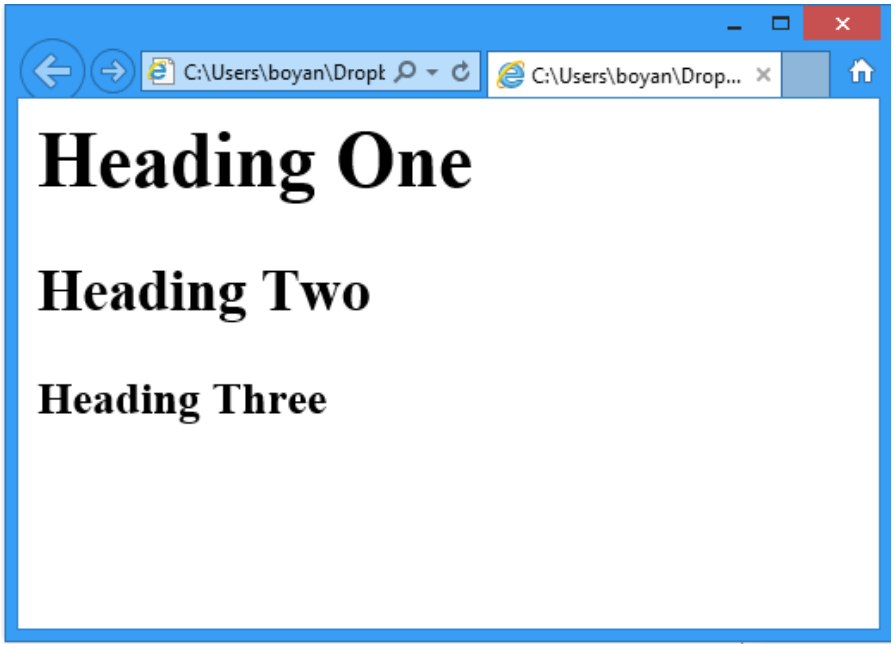
- ***Псевдо-класове***
  - A:**link** { color: red } /* unvisited link */
  - A:**visited** { color: blue } /* visited link */
  - A:**active** { color: green } /* link currently being pressed */
- ***Псевдо-елементи***
  - P:**first-line** { font-variant: small-caps; } /* The first-line pseudo-element will make the first line as formatted of a P element appear as if it were inside a P:first-line element, with the result effect on style*/
  - p:**first-child** { background: #ff0000 } /* sets a background color for the <p> element that is the **first** child of its parent */
  - p:**last-child** { background: #ff0000 } /* sets a background color for the <p> element that is the **last** child of its parent */
  - p:**first-of-type** { background: #ff0000 } /* sets a background color for the first <p> element of its parent */
  - P:**first-letter** { font-size: 200% } /* first-letter is as if it surrounds the "first letter" of the element. */
- ***Комбинации***
  - A.footnote:**visited** { color yellow } /* class and pseudo-class */
  - P.urgent:**first-line** { color red } /* class and pseudo-element */

CSS1 пример

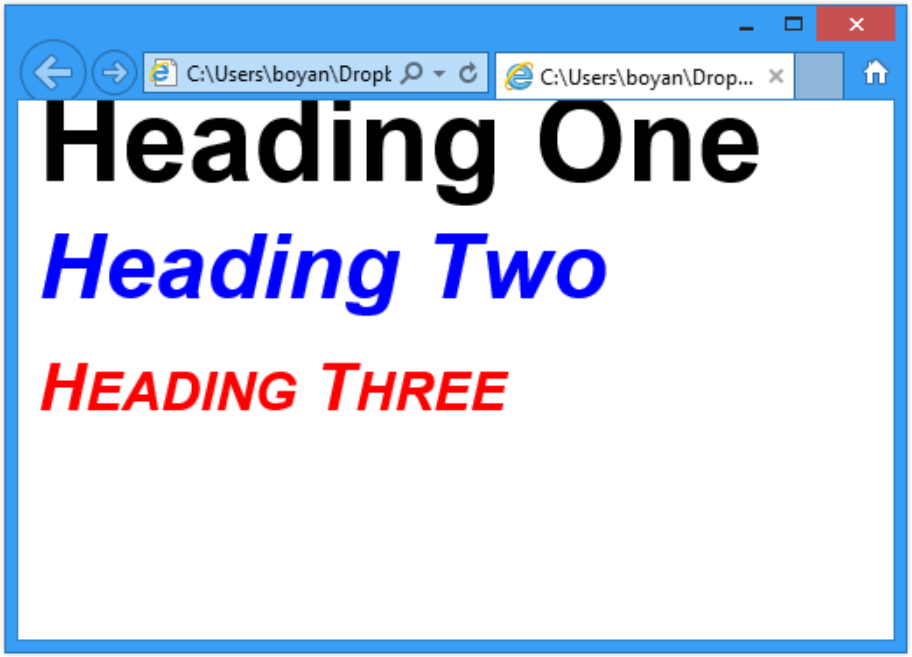
```
<html>
        <head>
                <STYLE TYPE="text/css">
h1, h2, h3 {
  font-family: helvetica;  font-weight: bold;
  font-size: 36pt;   line-height: 14pt;
  font-family: helvetica; font-variant: normal;}
h2 {font-size: 28pt; font-style: italic; color: blue; font-stretch: condensed}
h3 {font-size: 20pt; font-style: oblique; color: red; font-variant: small-caps}
                </STYLE>
        </head>
        <body>
                <h1> Heading One </h1>
                <h2> Heading Two </h2>
                <h3> Heading Three </h3>
  <body>
  </html>
```

CSS1 пример в браузър

HTML без CSS



HTML с CSS



Каскаден ред – клауза ! important

- При противоречиви правила (за стил на даден елемент) се избира правило по следния ред:
- **! important** правилата са с по-висок приоритет при еднаква тежест на дефинициите и съдържат клаузата **! important**. *Както авторът, така и читателят могат да задават important правила, затова в този случай авторските правила предефинират (отменят) тези на читателя*. Пример за използване на **! important**:
- **BODY { background: url(bar.gif) white; background-repeat: repeat-x ! important }**

•**repeat** Adding this tag just makes your background image tile like it does with the regular background tag.
•**no-repeat** Adding this tag makes your background image appear in it's regular size in the upper left hand corner of your page, the image shows only once.
•**repeat-y** Adding this tag makes your background image tile vertically on your page, along the left margin.
•**repeat-x** Adding this tag makes your background image tile horizontally on your page, along the top margin.

Каскаден ред – произход на правилата (Author's vs. Reader's)

- При конфликт, *авторските правила предефинират (отменят) тези на читателя.*
- *Авторските и читателските правила предефинират тези на браузъра.*
- Авторите трябва да използват **! important** правила много внимателно, за да не отменят някое важно читателско правило – например за цвят на текст върху даден фон, с цел добра четимост на текста.

Избор на правила по специфичност

- При конфликтни правила, избор на стил се прави и на база на специфичността на правилото.
- Колкото едно правило е по-специфично, толкова по-голям приоритет ще има то самото!
- Примери:
  - Брой на **ID** атрибутите в даден селектор.
  - Брой на **CLASS** атрибутите в даден селектор.
  - Брой на **HTML имена на елементи** в даден селектор.
  - Брой на псевдо-класове/елементи
  - "Pseudo-classes" and "pseudo-elements"

Още за специфичността

- Един селектор е по-специфичен от друг, ако:
- - има повече **ID** атрибути,
- - има повече **CLASS** атрибути,
- - има повече **имена на елементи**.

- Примери:
  - **UL UL** по-специфичен от **UL.**
  - **UL.urgent** по-специфичен от **UL UL UL UL ....**
  - **UL UL.urgent** по-специфичен от **UL.urgent.**

Cascading Style Sheets (CSS2)

- W3C Recommendation since 2001, http://www.w3.org/TR/REC-CSS2/
- CSS2 е изградена върху CSS1 и, с много малко изключения
- Всички валидни CSS1 стилови дефиниции са валидни CSS2 стилове.
- CSS2 поддържа специфични стилове за различни медии, така че авторите могат да представят техните документи за визуални браузъри, звукови устройства, принтери, брайлови устройства, преносими устройства, и др.

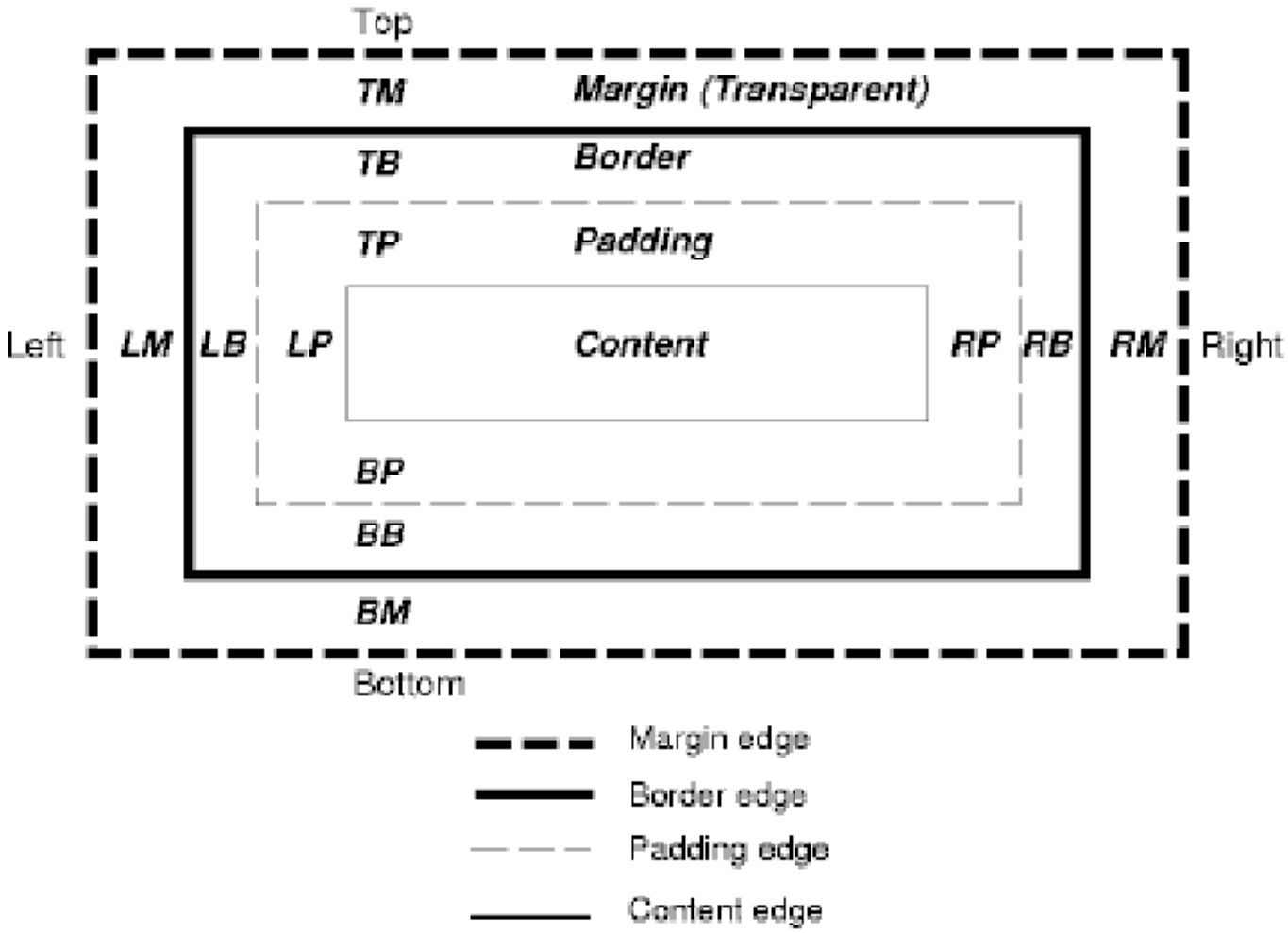
Cascading Style Sheets (CSS2)

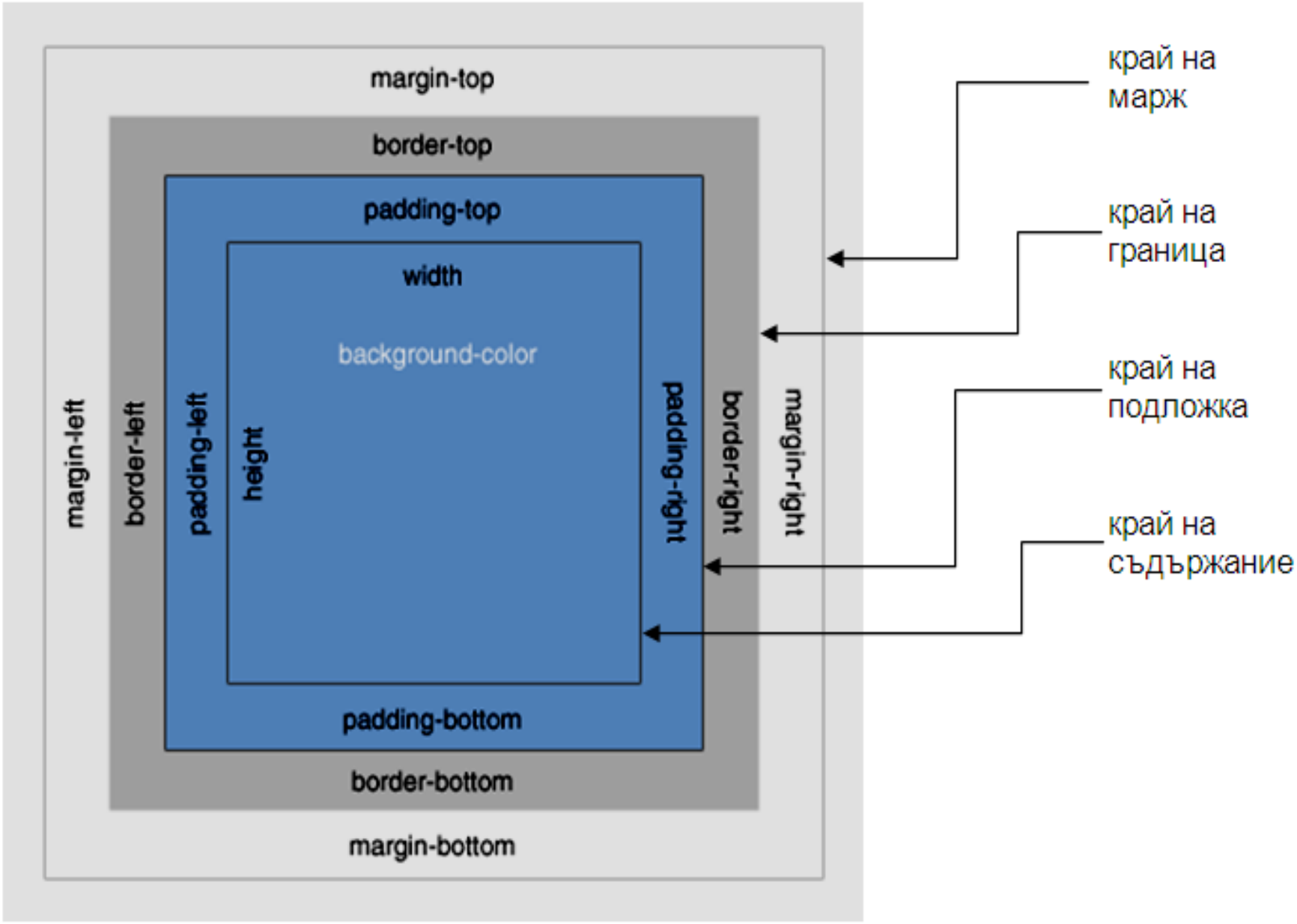
- Поддържа:
  - Различни медии
  - Схеми за позициониране на съдържанието
  - Зареждане на шрифтове (downloadable fonts)
  - Разполагане на таблици (table layout)
  - Интернационализация
  - Автоматично броене и номериране
  - Нови свойства на потр. интерфейс
- Работи както с XML, така и с HTML документи

Кутиен модел (усъвършенстван в CSS2)

- При показване на документ, CSS третира всеки елемент в документа като правоъгълна кутия
- Всяка кутия се състои от четири компонента:
  - съдържание
  - подложка (*padding*), заобикаляща съдържанието
  - граница (*border*), и
  - допустими граници или маржове (*margin*).
- Маржовете на всяка кутия са прозрачни, а границите й могат да имат стилове (непрекъсната или пунктирана линия).
- Подложката представлява зоната между границата и съдържанието. Цвят на фона се прилага към области вътре в рамката на границата, която включва подложката и съдържанието.

Елементи на CSS кутия: content, padding, border, and margin




margin-top
border-top
padding-top
width
background-color
height
margin-left
border-left
padding-left
padding-right
border-right
margin-right
padding-bottom
border-bottom
margin-bottom
край на
марж
край на
граница
край на
подложка
край на
съдържание

Блоково и поредово представяне 1/2

- Всяка кутия може да съдържа други кутии, отговарящи на елементи, които са вложени в нея.
- В CSS съществуват два основни вида кутии: блокови (*block*) и поредови (*inline*).
- Блоковите кутии представят съдържанието поблоково - всеки параграф се показва с пренасяне на нов ред преди и след параграфа
- Съдържанието на поредовите кутии може да се движи заедно с обкръжението си без пренасяне на нов ред, както например част от текста с удебелен шрифт в средата на параграфа.

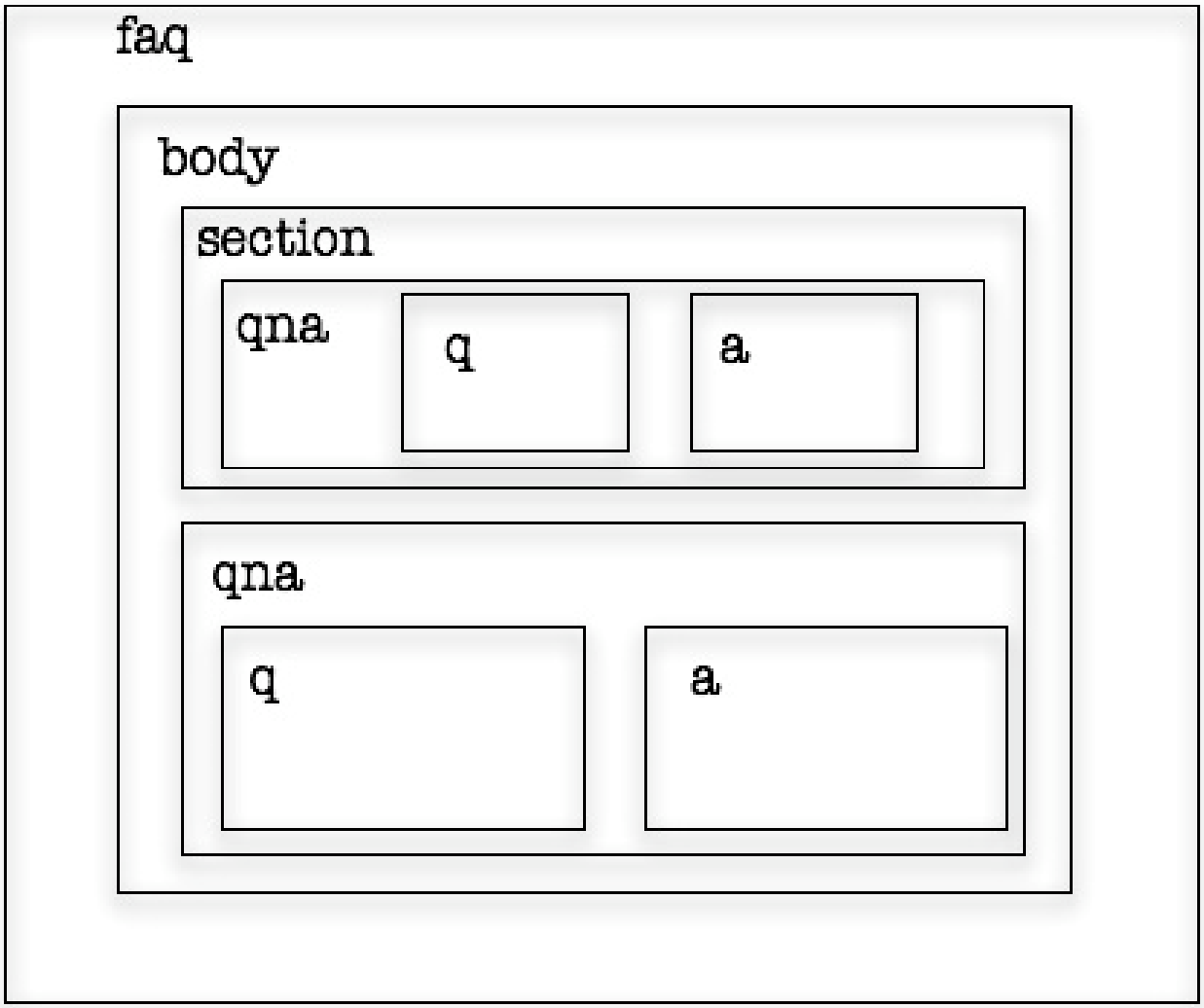
Блоково и поредово представяне 2/2

- В обичайното поточно представяне на съдържание, *блоковите кутии се нареждат една след друга вертикално*, докато *поредовите кутии се разполагат една след друга хоризонтално*. Например, параграфът е стек от блокови кутии-линии, всяка от които се състои от серия от поредови кутии.
- Има и други видове кутии, като например таблиците, които имат смесено представяне.
- В HTML или XHTML, блоковите кутии са създадени от елементи като например **<p>**, ** ** или **<table>**, докато поредовите кутии се създават от тагове като **<b>**, **** и ****, както и за съдържание като текст и изображения.

Свойство **Display**

Разлика между HTML и XML:

- HTML елементите са или block, или inline,
- при оформяне на XML с CSS браузърът не знае кои елементи трябва да се показват по блокове и кои поредово => използваме CSS за XML с **display** свойство, с допустими стойности:
  - **Block**
  - **Inline (по подразбиране!)**
  - **None (display:none ≠ visibility:hidden !)**
  - Inherit
  - Table
  - Table-row-group (equiv. to HTML element <TBODY>)
  - Table-row (equiv. to HTML element <TR>)
  - Table-cell (equiv. to HTML element <TD>)
  - Други …



Вграждане на кутии

CSS пример: “*rolodex.css*”

rolodex {display: block}

last {font-family: garamond; font-weight: bold; color: red}

name {display: block; font-family: garamond; text-decoration: italic; font-weight: bold; color: green; }

address {font-family: garamond; font-weight: bold; color: blue}

record { display:block; width:2in;
border-top: solid red;
border-right: solid red;
border-bottom: solid blue;
border-left: solid red }

CSS пример - XML

```
<?xml version="1.0"?>
<!DOCTYPE Rolodex SYSTEM "rolodex.dtd">
<?xml-stylesheet type="text/css" href="rolodex.css"?>
    <Rolodex>
      <record>
        <name gender = "female">
            <first>Jane</first>
            <last>Poe</last>
        </name>
.....
```

```
Rolodex {display: block}

last {font-family: garamond; font-weight: bold; color: red}

name {display: block; font-family: arial; text-decoration: underline; color: green; }

address {font-family: times; font-style: italic; color: blue}

record {   display:block; width:2in;
           border-top: solid red;
           border-right: solid red;
           border-bottom: solid blue;
           border-left: solid red }
```

xml10_example3.xml - Notepad

File Edit Format View Help

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/css" href="exampl
<Rolodex>
  <record>
    <name gender = "female">
        <first>Jane</first>
        <last>Poe</last>
    </name>
    <address>
        Av. St. Clair on the Heavens 1643, Br
    </address>
  </record>
  <record>
    <name gender = "male">
        <first>Jim</first>
        <last>Hackman</last>
  </name>
    <address>
      Balsha 6 St., Sofia 1643, Bulgaia
```

C:\Boyan\Lectures\2004_5\XML\lessons\xml10_ex...

File Edit View Favorites Tools Help

Back Search Favorites Media

Address \2004_5\XML\lessons\xml10_example3.xml Go Links

Google

Jane **Poe**
Av. St. Clair on the Heavens 1643, Bristol, MA 2131, USA

Jim **Hackman**
Balsha 6 St., Sofia 1643, Bulgaia

Done My Computer

Позициониране в CSS

- След като съдържанието на всеки елемент може да се показва като кутия, процесът на оформление на представянето се свежда до вземане на решение кой тип кутия (с блоково или поредово представяне) да се използва за всеки елемент в документа.
- Освен това, трябва да се определи местоположението (тоест позиционирането) на тази кутия.
- Спецификацията на CSS 2 дефинира чрез стойността на свойството **position** няколко вида на позициониране - нормалното (статично) позициониране на потока, относително, абсолютно, плаващо и фиксирано позициониране.

Схеми за позициониране 1/3

- **Статично позициониране - position:static** задава статично позициониране. То се контролира от браузера и е най-бързо.
- **Относително позициониране** - подобно на статичното, като обаче горната и лявата позиции могат да бъдат предефирани от CSS. То е вариант на нормалния поток и се задава със стойност **position: relative**.
- https://developer.mozilla.org/en-US/docs/Web/CSS/position

Схеми за позициониране 2/3

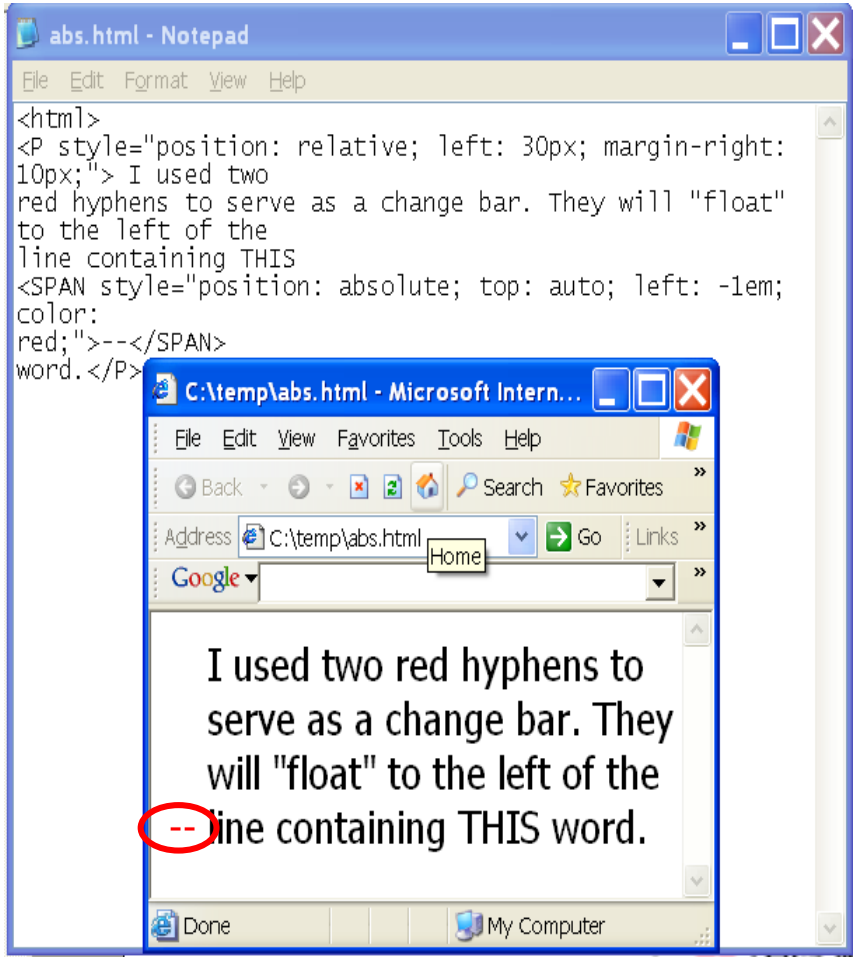
- **Абсолютно позициониране (position: absolute)** имаме тогава, когато на кутията е присвоено фиксирано отместване от местоположението ѝ, определено спрямо фиксирания поток. То не оказва влияние върху това как са определени Sibling кутиите.
- В зависимост от нивото на вложеност, абсолютно позиционираният елемент може да закрие други елементи.
- https://developer.mozilla.org/en-US/docs/Web/CSS/position

Пример за относително с абсолютно позициониране

<P style="position: relative; left: 10px; margin-right: 10px;"> I used two red hyphens to serve as a change bar. They will "float" to the left of the line containing

<SPAN style="position: absolute; top: auto; left: -1em; color: red;">--</SPAN> THIS word.</P>

- Резултат →



Друг пример

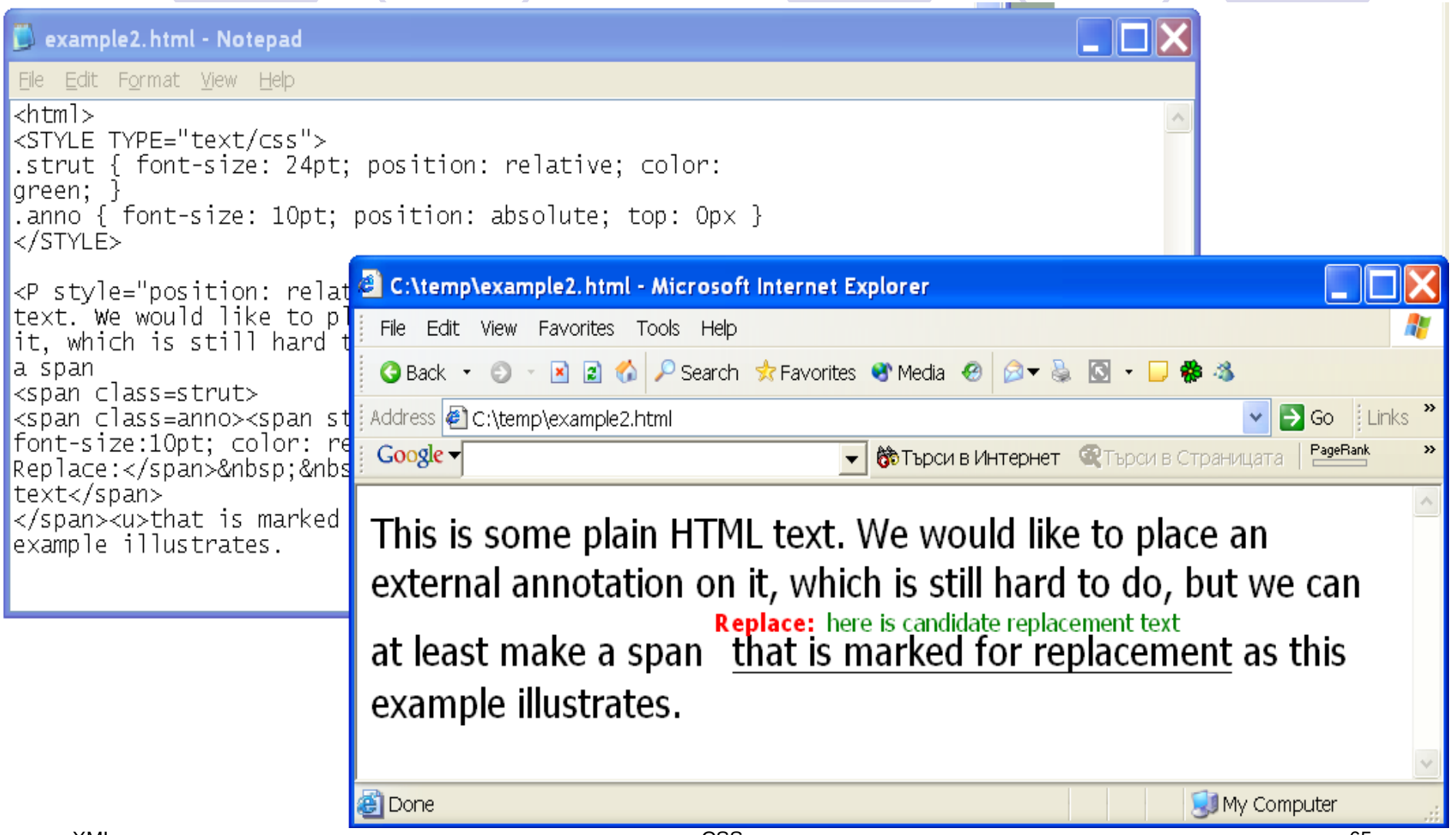
Използваме стила

```
<STYLE TYPE="text/css">
.strut { font-size: 24pt; position: relative; color: green; }
.anno { font-size: 10pt; position: absolute; top: 0px }
</STYLE>
```

в примера

```
<P style="position: relative">This is some plain HTML text. We would like to place an external annotation on it, which is still hard to do, but we can at least make a span
<span class=strut>
  <span class=anno>
        <span style="font-weight: bold; font-size:10pt; color: red">
              Replace:
  </span>
        here is candidate replacement text
  </span>
</span><u>that is marked for replacement</u> as this example illustrates.
```

Резултат


example2.html - Notepad
File Edit Format View Help
<html>
<STYLE TYPE="text/css">
.strut { font-size: 24pt; position: relative; color:
green; }
.anno { font-size: 10pt; position: absolute; top: 0px }
</STYLE>
<P style="position: relat
text. We would like to pl
it, which is still hard t
a span

<span st
font-size:10pt; color: re
Replace: &nbs
text
<u>that is marked
example illustrates.
C:\temp\example2.html - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address C:\temp\example2.html
Go Links
Google
Търси в Интернет
Търси в Страницата
PageRank
This is some plain HTML text. We would like to place an external annotation on it, which is still hard to do, but we can at least make a span
Replace: here is candidate replacement text
that is marked for replacement as this example illustrates.
Done
My Computer

Схеми за позициониране 3/3

- **Плаващо позициониране -** със стойност **position:float** - създава кутия, като позволява друго съдържание да тече (плава) около тази кутия - https://developer.mozilla.org/en-US/docs/Web/CSS/float .
- **Фиксирано позициониране** - подкатегория на абсолютното позициониране. Елемент, за който имаме **position:sticky**, винаги има изгледа като съдържащия блок. За непрекъснати медии, като например компютърен екран, фиксираните елементи няма да се движат, когато документът се превърта (скролира). За пейджър медиите, фиксираният елемент ще се повтаря на всяка страница - https://developer.mozilla.org/en-US/docs/Web/CSS/position.

Още примери

- **W3 School: CSS School**
  - **http://www.w3schools.com/css/**
  - **CSS Basic:**
    - **CSS Introduction**
    - **CSS Syntax**
    - **CSS How To**
    - **CSS Background**
    - **CSS Text**
    - **CSS Font**
    - **CSS Border**
    - **CSS Margin**
    - **CSS Padding**
    - **CSS List**

- CSS Advanced:

- **•CSS Dimension**
- **•CSS Classification**
- **•CSS Positioning**
- **•CSS Pseudo-class**
- **•CSS Pseudo-element**
- **•CSS Media Types**


?
Q
&
A
!