web разработка

Рязане на сайтове част 2
Позициониране

Съдържание

- CSS optimization
- Нулиране настройките на браузърите
- CSS-pseudo-classes
- Float design
- Responsive design
- Viewport

CSS optimization

CSS optimization

Как нашия CSS да работи по-бързо?

color: #fff

вместо color:white - за да не се налага CSS да “превежда” white в код

или color:#ffffff - по-малко знаци.

padding-left:12px

padding-right:12px

вместо padding: 0 12px; - за да не се извършват излишни операции и задаване на ст-сти за top/bottom

Img-sprites вместо много картинки за фон

CSS optimization

- Задаваме стиловите правила като се движим от общото към конкретното - т.е -
  Първо задаваме правилата, важащи за всички или за повечето елементир след това задаваме конкретните свойства за отделните елементи
- Обединяваме правилата, важащи за един селектор.

- Разделяме файла на зони, като оставяме коментари, коя част от стиловия файл към кои елементи се отнася.

Нулиране настройките на браузърите

Нулиране настройките на браузърите

Изброяване на всички елементи в страницата – повече писане, но имаме контрол върху елементите или

Универсален селектор - * - по-малко писане, но са възможни неочаквани резултати

Ресурси

Нулиране настройките на браузърите

… so this method /universal selector/ should never be used unless you know exactly what to expect…

CSS

Pseudo-classes

CSS pseudo-classes

http://www.w3schools.com/cssref/css_selectors.asp

```
selector:pseudo-class {
        property:value;
}
```

CSS pseudo-classes - 2

Примери -

```
1//     a:hover {
            color: #fof;
       }

2//     p:first-child {
            color: blue;
       }

3//     p:first-of-type {
            background: #f00;
       }
```

```
p:nth-child(2) {
            background: #f00;
       }

p:nth-of-type(2) {
            background: #f00;
       }

p:last-child {
            background: #f00;
       }

p:last-of-type {
            background: #f00;
       }
```

Float design

display: ...

Позициониране чрез

display: inline - div или други блокови елементи имат поведение на inline елементи

display: block - span или други inline елементи имат поведение на block елементи

display: inline-block - комбинирано поведение на inline и block елементи - елементите се подреждат като , но вътре са с поведение на

display: none - елемента не се визуализира **/падащи менюта!/**

Float design - 1

float *позициониране изкарва елемента от стандартната подредба*

Приема стойности left и right

!!! Когато елементите са float-позиционирани, може да фиксирате ширината им, но височината се определя от тяхното съдържание.
Когато височината се определя от съдържанието, background може да не бъде приложен правилно, елемента може да излезе извън рамките на родителския елемент. Използваме **clear.**

Float design - 2

clear • clear контролира поведението на float елементите /прекъсва float/

стойности: left / right / both / none

```
#footer{
    clear: both;
    }
```

Float позициониране - https://csstricks.com/all - about -floats/

Float позициониране с много примери - http://alistapart.com/article/css-floats-101

Float design - 3

Проблем с преливащо съдържание / Преливане по височина и по ширина

2 начина за оправяне:

1//

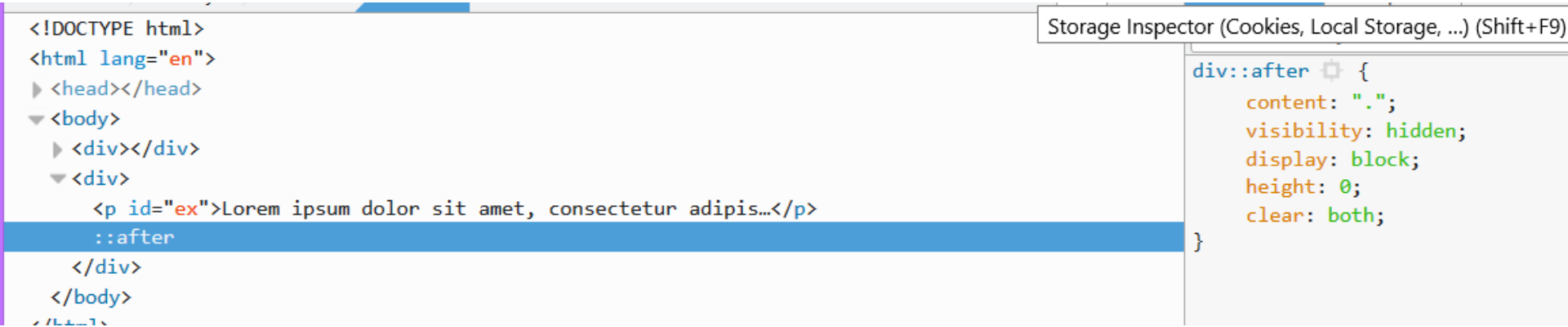
```
#id1{
    overflow: auto;
    }
```

2// на родителския елемент

```
#id1:after{
    content: ".";
        display: block;
        height: 0;
        clear: both;
        visibility: hidden;
    }
```

Float design - 4

::after се слага на родителския елемент, за да се изчисти преливането от float



Задачи -

позиционирайте менюто и другите елементи в първата и втората част от psd файла.

Responsive design

Responsive design

Responsive design /адаптивен дизайн/ е подходът визуализацията на уеб сайта на различни устройтва да бъде съобразена/да се променя съобразно особеностите на устройствата - Resolution, DPI, color range, etc…
Стремим се към по-добър (:)) UX на мобилните устройства, настолни компютри, телевизори ...

Responsive design - 3

Fluid layout - пропорционални стойности /в %/ за width, margin, etc…

Flexible types - всички големини на шрифтове се базират на основен елемент /root/

Flexible images - променят размерите си, но не излизат от размера на контейнера си.

Media queries - прилагат се стилове съобразени с конкретно устройство.

Responsive design - 3

Fluid layout - пропорционални стойности /в %/ за width, margin, etc…

+ Отговаря по-добре на резолюцията на съответното устройство и спестява кода за media queries
- Получават се по-големи бели пространства при големи екрани - (TV)
- Чупи се, когато се използват елементи с фиксирани размери - изображенията, които по принцип са с фиксиран размер.

Например -

резолюция: 1024px,

Контейнер с width: 60% (= 60% * 1024 = 614.4px)

И изображение с width: 500px

При резолюция 780px, ширината на контейнера е все още (= 468px), ширината на изображението е 500px. Изображението излиза извън контейнера. Решаваме проблема с max-width, което е по-силно свойство от width и ако ширината на изображението е по-голяма от тази на контейнера, то заема ширината на контейнера.

Responsive design - 4

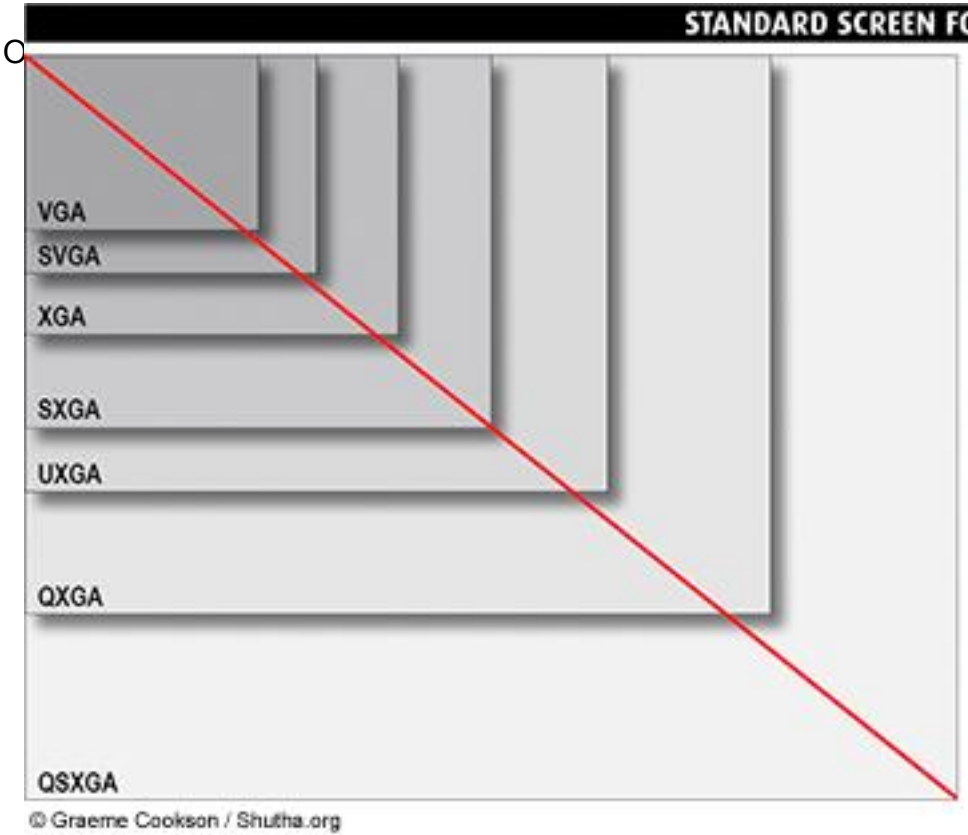
Flexible types - всички големини на шрифтове се базират на родителски елемент или root-елемент.

Големина, базирана на родителски елемент - вместо px използваме пропорционални единици - em
Ако искаме да изчислим големината, която искаме да постигнем -
em = target / root
1.4375em = 23px / 16px

```
body{ font-size:16px;}
 body header{ font-size:23px;} // 23 / 16 = 1.4375em
```

```
body{ font-size:16px;}
 body header{ font-size:1.4375em;}
```

Responsive design - екрани



Name	Size in Pixels	Ratio	Year
CGA	320 x 200	8:5	1981
EGA	640 x 350	8:5	1984
VGA	640 x 480	4:3	1987
SVGA	800 x 600	4:3	1990
XGA	1024 x 768	4:3	1990
SXGA	1280 x 1024	5:4	1994
UXGA	1600 x 1200	4:3	1996
QXGA	2048 x 1536	4:3	since 2000
QSXGA	2560 x 2048	5:4	

Responsive design - 5

Чрез **@media queries** задаваме различен дизайн за екрани с различен размер /презаписваме правилата за всеки размер/

```
@media only screen and (max-width: 320px){
        #header{
        width: 98%; }
        }
```

print - при принтиране

screen - за екрани

speech - за електронни четци

all - за всички типове

Responsive design - 6

Условия в **@media queries**

max - width
min - width
max/min - height
max/min - Color

Може да се задават и по няколко условия:
@media only screen and (max-width:960px) and (min-width:481px)

@media queries - особености

Условията се изпълняват отгоре надолу, така че последователността им има значение!

Дизайните по - надолу презаписват по - горните.
Т.е. ако сложим min - width:
320px най - долу, той ще презапише всички останали

http://www.w3schools.com/cssref/css3_pr_mediaquery.asp

Responsive design - 7

Media queries - поддържат се във всички по-големи браузъри.

Пример ***!!! всяко по-долно правило презаписва горното!!***
!!!как посочваме размерите на устройствата!!!

```
.box {width: 250px; height: 250px; display: inline-block}
@media only screen and (max-width: 1024px) {
 .box {width: 300px; height: 300px; }
}
@media only screen and (max-width: 960px) {
 .box {width: 310px; height: 310px;}
}
@media only screen and (max-width: 480px) {
 .box { display: block; width: 95%; height}
```

Responsive design - 8

Можем да ползваме различни CSS файлове за различните размери екран

```
<head>
    <link rel="stylesheet" media="(min-width:320px)" href="min-320px.css">
    <link rel="stylesheet" media="(min-width:640px)" href="min-640px.css">
...
</head>
```

responsive design/адаптивен дизайн - упражнения и материали

Viewport meta tag

Viewport meta tag

Viewport определя видимата част от уеб страница според големината на екрана

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

Задължителен мета таг - устройства с еднакви размери, но различна резолюция.

повече информация

set the viewport