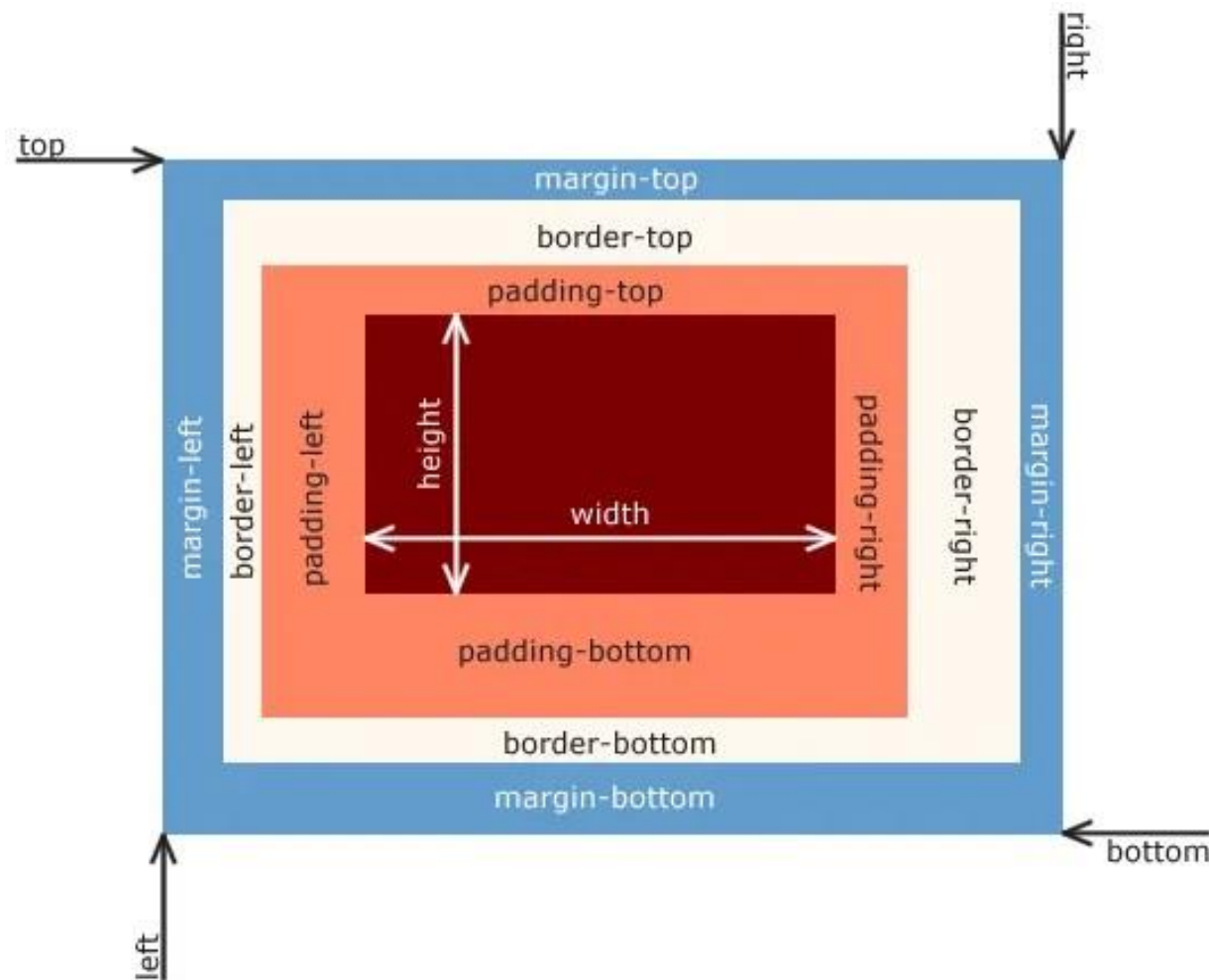


The background features abstract, overlapping green geometric shapes, primarily triangles and polygons, in various shades of green, creating a modern and dynamic visual effect.

Front-End Web Development CSS Box Model

- **CSS Box Model** е инструмент за проектиране на по-сложни оформления и позициониране на елементи на уеб страница или за подравняването им с други. Всички елементи могат да се разглеждат от гледна точка на box model концепцията.



► **Област на съдържанието** - текст, изображение, видеоплейър и т.н.

- width, min-width, max-width, height, min-height, max-height.

► **Padding зоната** се дефинира с padding-top, padding-right, padding-bottom, padding-left, padding. Padding-block и padding-inline са много подобни на padding-top, padding-bottom и padding-left, padding-right, но свойствата padding-block и padding-inline зависят от посоките на block и inline - padding-block-start, padding-block-end и padding-inline-start, padding-inline-end. Свързани тясно с writing-mode и direction.

► **Border зоната** (border-width border-style border-color)

- border-width: medium | thin | thick | length |

- border-style: none | hidden | dotted | dashed | solid | double | groove | ridge | inset | outset

- border-top-width border-top-style border-top-color border-top-left-radius border-top-right-radius

- border-bottom, border-left, border-right, border-inline, border-block, border-radius, border-image

► **Margin зоната** включва празна област, използвана за отделяне на елемента от съседните му - width, height, margin-top, margin-right, margin-bottom, margin-left, margin. CSS свойствата margin-block и margin-inline са подобни margin-top, margin-bottom и margin-left, margin-right, но свойствата margin-block и margin-inline зависят от посоките на блока и inline, за това - margin-block-start, margin-block-end и margin-inline-start, margin-inline-end.

► **outline**: outline-width outline-style outline-color | initial | inherit; outline-offset

- Свойството за оразмеряване на кутията **box-sizing** контролира как моделът на кутията изчислява своя размер.
- **box-sizing**, зададен с **content-box** (по подразбиране): свойствата за ширина и височина включват съдържанието, border и padding.

- Width width + padding-left + padding-right + border-left + border-right
- Height height + padding-top + padding-bottom + border-top + border-bottom

```
#example1 { box-sizing: content-box;  
             width: 200px;  
             height: 100px;  
             padding: 30px;  
             border: 20px solid blue; }
```

- **box-sizing**, зададен с **border-box**: свойствата за ширина и височина (и свойства за мин./макс.) не включват border и padding в изчисляването им. Целият елемент е с размерите width: 200px; и height: 100px, не зависимо, че има padding и border. Само margins е с друг статут.

```
#example2 { box-sizing: border-box;  
            width: 200px;  
            height: 100px;  
            padding: 30px;  
            border: 20px solid blue; }
```

► **block-level / блокови елементи**

<address><article><aside><blockquote><canvas><dd><div><dl><dt><fieldset>
<figcaption><figure><footer><form><h1><h6><header><hr><main><nav>
<noscript><p><pre><section><table><tfoot><video>

- Блоковите елементи генерират кутия, която се позиционира като блок (започва на нов ред, завършва с нов ред).

► **inline / поредови елементи**

<a><abbr><acronym><bdo><big>
<button><cite><code><dfn><i>
<input><kbd><label><map><object><output><q><samp><script><select><small>
<sub><sup><textarea><time><tt><var>

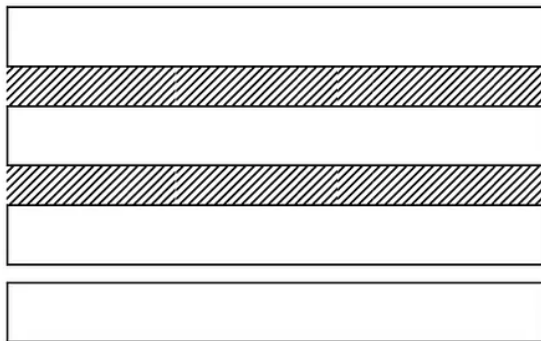
- Поредовите елементи генерират кутия, която се изобразява в рамките на текущия ред и заема място, колкото е нужно на съдържанието.

► Типът определя как кутията се държи по отношение на потока на страницата и по отношение на други кутии на страницата.

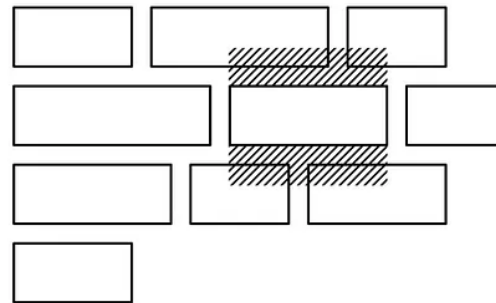
Изключения се правят с **display**. Възможните стойности в този контекст са:

- inline (показва елементите един след друг и игнорира стойностите за размер)
- block (елементът се изобразява като блок, започва на нов ред / вертикалности заема цялата ширина, отстъпите се препокриват)
- none (скрива елемента и не запазва мястото)

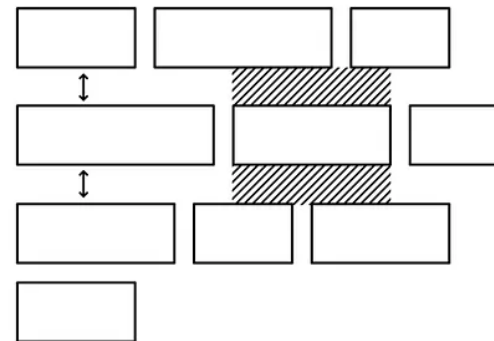
block



inline



inline-block



The background features abstract, overlapping green geometric shapes, primarily triangles and polygons, in various shades of green, creating a modern and dynamic visual effect.

Front-End Web Development CSS Layout - Position

- Позиционирането включва не само задаване на точка от полето на страницата, в която или към която определен елемент ще бъде свързан, а и параметри, чрез които се определя кои негови части ще покриват частично или изцяло части на друг елемент, кои части ще бъдат скрити, кои ще бъдат прозрачни.

Схеми за позициониране

- **Нормален поток:**

- Контекстно позициониране - генерираните от елементите кутии се позиционират в зависимост от контекста (блоков или поредов, но не и двата едновременно). Всеки елемент се опитва да се намести максимално нагоре и максимално наляво, ако не се приложат изключенията.

- **Плаваща (float) схема:** float + clear

- Кутията първоначално се позиционира според нормалната потокова схема, а след това се отмества наляво / надясно в зависимост от указаната стойност.

float: none | left | right | initial | inherit;

- Свойството clear контролира потока до плаващите елементи, указва какво трябва да се случи с елемента, който е до плаващ елемент.

clear: none | left | right | both | initial | inherit;

- **Абсолютно позициониране:**

- Кутията изцяло се премахва от нормалния поток и се позиционира според зададените координати спрямо съдържащия я блок.

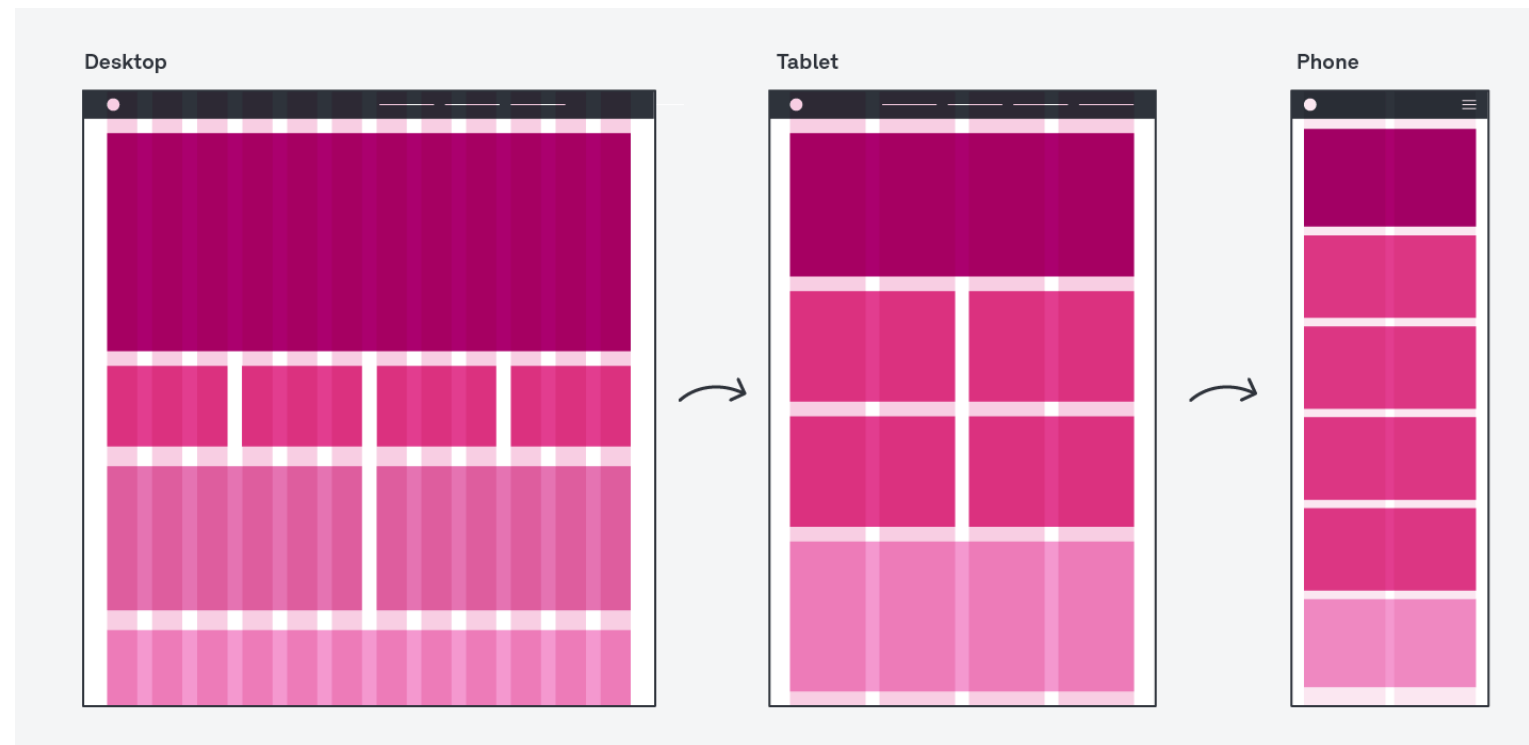
► **Стиловите формати за позициониране** се управляват по-прецизно и става възможно динамичното преместване на графични компоненти по полето на страницата.

- position: static | absolute | fixed | relative | sticky | initial | inherit
- top, left, right, bottom: auto | length px, %, отрицателни стойности | initial | inherit
- **position: static** - винаги се позиционира според нормалния поток на страницата. Елементите не се влияят от свойствата отгоре, отдолу, отляво и отдясно.
- **position: relative** - се позиционира спрямо текущото си положение. Свойствата top, right, bottom и left се използват за позициониране на елемента. Няма пренареждане на други елементи, застава над тях.
- **position: absolute** - позицията се определя спрямо указани абсолютни координати. За да действа, трябва да има родителски елемент от тип relative. Свойствата top, right, bottom и left се използват за позициониране на елемента спрямо родителския елемент.
- **position: fixed** - елементът е позициониран спрямо прозореца за изглед, което означава, че винаги остава на едно и също място, дори ако страницата се превърта. Свойствата top, right, bottom и left се използват за позициониране на елемента.
- **position: sticky** - елементът превключва между относителен и фиксиран в зависимост от позицията на превъртане. Позиционира се относително, докато се срещне дадена позиция на отместване в прозореца за изглед - след това "залепва" на място (като позиция: фиксирана) - position: -webkit-sticky; position: sticky; top: 0;

- ▶ **z-index**: N; - задава поредния номер (N) на слоя в стека (и отрицателни стойности). Използва се при управление на превключването между различни слоеве.
- ▶ **visibility**: visible | hidden | collapse | initial | inherit; - атрибут за управление на видимостта на слоя. Винаги заемат мястото си, дори и скрити. collapse - тази стойност премахва ред или колона, но не засяга оформлението на таблицата. Унаследяване на атрибутите от родителския слой се установява чрез inherit. (за разлика от display:none, където изцяло изчезва елемента)
- ▶ **clip**: rect(left, top, right, bottom); - определя размерите на правоъгълна видима част от слоя. Размерите се задават в пиксели спрямо горния ляв ъгъл на слоя. Ако този параметър не е зададен, се приемат стойностите указани в width и height. Ако и те са изпуснати, горният ляв ъгъл на слоя се определя от неговата дефиниция, а размерът - от обема на съдържанието. clip-path - circle, ellipse, polygon
- ▶ **aspect-ratio** (number/number) дефинира съотношението между ширината и височината на даден елемент. Подходящо свойство в респонсив дизайн, където елементите често варират по размер и трябва да се запази съотношението между ширина и височина.

The background features abstract, overlapping green geometric shapes, primarily triangles and polygons, in various shades of green, creating a modern and dynamic visual effect.

Front-End Web Development Responsive Grids



- ▶ **Responsive Grids** - отзивчивите мрежи / решетки са мощен инструмент за създаване на елегантен уеб дизайн, който се адаптира към различни размери на екрана, без да се изкривява. Уеб сайтът става лесен за навигиране при различните случаи. Ускорява се времето за зареждане на страницата, като елиминират ненужния код от множество версии на едно и също оформление. Съдържанието може да бъде разделено на модули, което прави много по-лесно създаването на динамични уеб сайтове, без притеснение за съвместимостта в различни браузъри и устройства.

Три основни типа адаптивни мрежови системи

- ▶ **Фиксираната мрежа** се състои от колони, които имат постоянна ширина, независимо от размера на екрана, на който се гледат. Помага за осигуряване на съгласуваност между устройствата.
- ▶ **Плавната мрежа** - fluid grid - използва базирани на проценти ширини вместо фиксирани ширини, което означава, че колоните ще се преоразмеряват автоматично в зависимост от размера на екрана, на който се гледат. Позволява повече гъвкавост, тъй като уебсайтовете се адаптират към променящите се размери.
- ▶ **Хибридната мрежа** съчетава както фиксирани, така и плавни елементи, за да създаде оптимално потребителско изживяване на всички устройства и екрани.

► Columns

Колоните са най-същественият елемент на всяка отзивчива мрежова система. Колоните разделят страницата на секции, което улеснява потребителите при бързо сканиране на съдържание и намиране на конкретна информация. Броят на колоните, използвани в оформлението, ще зависи от размера и сложността на проекта, но обикновено варира от две до дванадесет. Елементите вътре в набора от колони са организирани и последователни.

► Gutters

Невидимите линии са пространството между колоните или редовете в дизайна. Те са важна част от мрежите, осигуряващи визуално разделяне между различните елементи, като същевременно придават на страницата сплотен вид и усещане. Осигуряват също баланс на дизайна, като създават равномерен поток и установяват връзки между елементите.

► Margins

Полетата помагат за установяване на визуална йерархия и гарантират, че елементите не се припокриват ненужно. Чрез добавяне на празно пространство между елементите полетата улесняват потребителите да правят разлика между различните секции на страницата, без да бъдат затрупани от твърде много информация наведнъж.

► Breakpoints

Точките на прекъсване позволяват да се оптимизира дизайна за различни размери на екрана. По същество точките на прекъсване се отнасят до конкретните точки, където оформлението на уебсайт се променя. Те се задействат, когато са изпълнени определени критерии (напр. определени размери на екрана) и помагат да се гарантира, че съдържанието се показва правилно, без значение как потребителят осъществява достъп до него.

► Fields

Полетата се отнасят до „кутиите“, които се появяват в мрежата. Колко са и къде са поставени - ще диктува типовете съдържание, което може да се използва на дадена страница, което ги прави изключително полезен инструмент за по-сложни уеб страници или оформления.

- ▶ Стъпка 1: Определяне на целевия размер на екрана. За какъв размер трябва да бъде оптимизиран дизайнът? Какви са минималните и максималните размери? Тук се решава какъв вид оформление на мрежата трябва да се използва, както и кои елементи трябва да се показват на различни точки на прекъсване.
- ▶ Стъпка 2: Избиране на оформление и извършване на измервания. Колко колони, колко широка трябва да бъде всяка колона? Ще има ли допълнителни колони при по-големи точки на прекъсване? Има ли специфични измервания, които трябва да се вземат предвид при проектирането на страницата? Броят на колоните ще зависи от целта и структурата на проектирания сайт, но обикновено наличието на повече от 12 колони може да направи нещата да изглеждат претрупани и трудни за четене. Ако е възможно, нека са до 8-10, тъй като това позволява повече гъвкавост при проектирането. Освен това, използването на брой колони, делими на четири (напр. 4, 8, 12), улеснява равномерното разпределяне на елементи във всяка страница или раздел на уебсайта.
- ▶ Стъпка 3: Конфигуриране на ширините на gutters и полетата. Използват се, за да осигурят достатъчно място между елементите, така че всичко да изглежда последователно, без значение на какъв размер устройство се гледа. Колкото по-тесни са gutters, толкова повече се усеща, че всички полеви елементи са част от една група с безпроблемен поток между тях. От друга страна, по-широките създават усещането, че всяко поле принадлежи към отделни обекти или категории.
- ▶ Стъпка 4: Изграждане на мрежата. Това включва създаване на класове за всеки елемент на страницата (напр. header, footer, sidebars, main), задаване на ширини и височини и задаване на полета и paddings, където е необходимо. Вземат се предвид мобилните устройства при проектирането за лаптоп или настолен компютър - дефиниране на точки на прекъсване.
- ▶ Стъпка 5: Тестване на различни размери на екрана.

The background features abstract, overlapping green geometric shapes, primarily triangles and polygons, in various shades of green, creating a modern and dynamic visual effect.

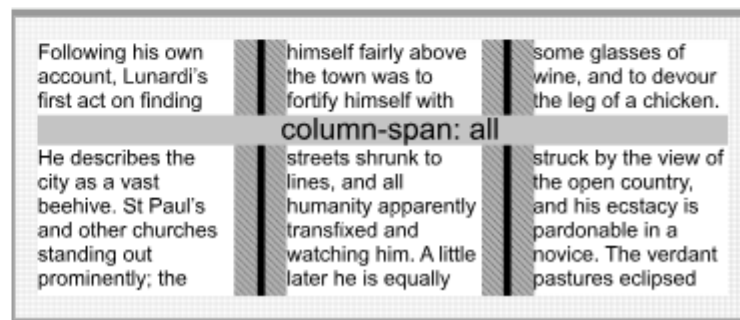
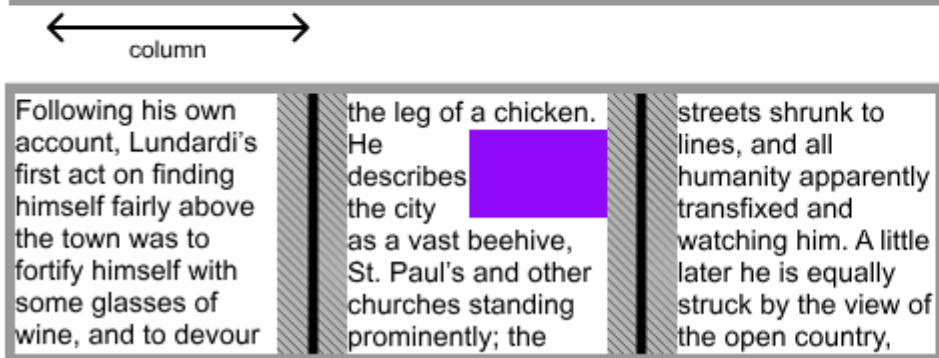
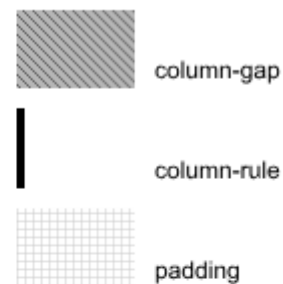
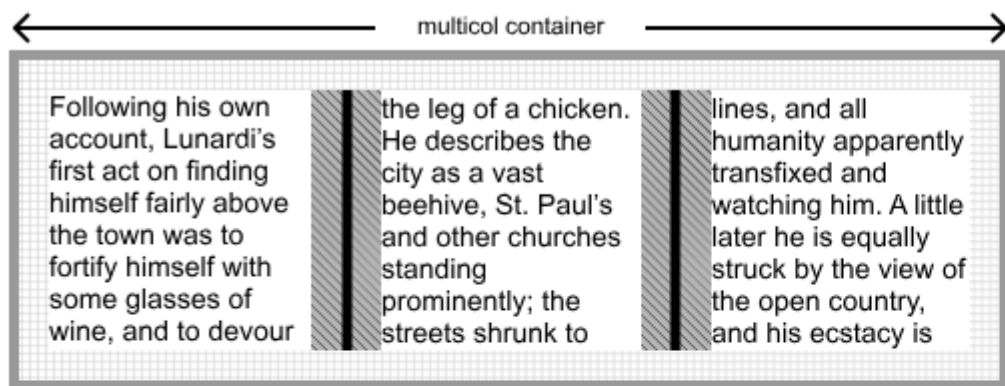
Front-End Web Development Multi-column Layout

- ▶ Основната идея на **Multi-column** е, че се взема част от съдържанието и се прелива в множество колони, както във вестник. Използва се едно от двете свойства: Колоните трябва да бъдат зададени или чрез свойството `column-width` или чрез `column-count`, но никога не се използват заедно. Свойството `column-count` указва броя на колоните, в които да се разпредели съдържанието. Свойството `column-width` указва идеалната ширина, оставяйки на брауъра да разбере колко колони ще се поберат.
- ▶ Няма значение кои елементи са вътре в съдържанието, което се превръща в многоколонен контейнер, всичко остава в нормален поток, но разделено на колони.

- ▶ Пример с `column-width`:

```
.container { max-width: 800px; margin: 0 auto; column-width: 14em;}
```

Брауърът присвоява толкова 14em колони, колкото могат да се поберат, и след това споделя оставащото пространство между колоните. Не е необходимо да се добавят медийни заявки и да се променя броя на колоните за различни точки на прекъсване, вместо това се определя оптимална ширина и брауърът ще я изработи.



- ▶ Стилизиране при **Multi-column**: Не могат да се таргетират отделните колони или да се прилага JavaScript.
- ▶ column-rule-color, column-rule-style, column-rule-width, column-span, column-width, column-fill, columns

```
.container { max-width: 800px; margin: 0 auto; column-width: 14em; column-gap: 2em; column-rule: 1px solid #ccc;}
```

- ▶ column-span: all; - елементът се разпростира върху всички полетата на колоните.
- ▶ break-inside - контролиране на прекъсването на съдържанието. С break-inside браузърът избягва прекъсвания в изображения, кодови фрагменти, таблици и списъци.

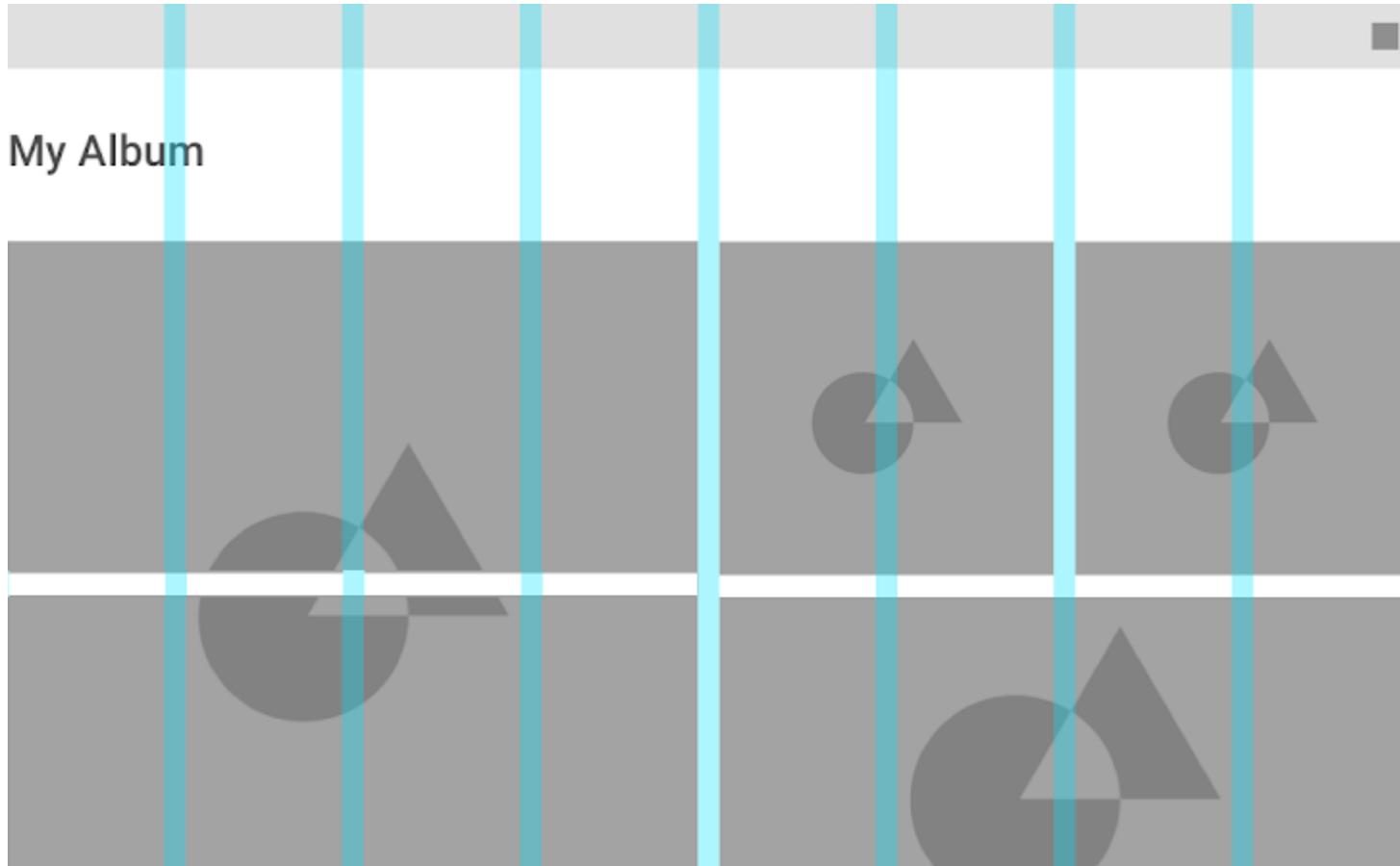
```
break-inside: auto | all | always | avoid | avoid-column | avoid-page | avoid-region | column | left | page | recto | region | right | verso | initial | inherit;
```

- ▶ Свойството break-after указва дали след посочения елемент трябва да се появи прекъсване на страница, колона или регион.
- ▶ Свойството break-before указва дали прекъсване на страница, колона или регион трябва да се появи преди посочения елемент.
- ▶ Multi-column също могат да се използват като резервен вариант за Grid и Flex оформление. Ако посочите едно от свойствата на контейнер, след това превърнете този контейнер във Flex или Grid оформление като използвате display: flex или display: grid, всяко поведение на колона ще бъде премахнато. Браузърите, които не поддържат Grid, ще получат многоцветен дисплей. Тези, които поддържат Grid, ще получат Grid Layout.

The background features abstract, overlapping green geometric shapes, primarily triangles and polygons, in various shades of green, creating a modern and dynamic visual effect.

Front-End Web Development Responsive Web Design Grid-View

My Album



► **Grid-View** (мрежов изглед) се прилага за повече контрол върху уеб страницата.

Съдържанието е разделено в колони, обикновено до 12 колони, има обща ширина от 100% и ще се свива и разширява с преоразмеряването на прозореца на браузъра.

- 12 колони се разделят добре на по три, четири и шест колони, което е достатъчно гъвкаво, за да се справи доста лесно с всякакви видове съдържание. Освен това може да се направят симетрични или асиметрични, което дава много възможности да се надхвърли обикновената симетрия.

► https://www.w3schools.com/css/css_rwd_grid.asp

Изграждане: Изписва се `<meta name="viewport" content="width=device-width, initial-scale=1.0">`


Присвоява се `* { box-sizing: border-box; }` (за всички елементи)

Прави се подсигуряване с `.row::after { content: ""; clear: both; display: table; }`

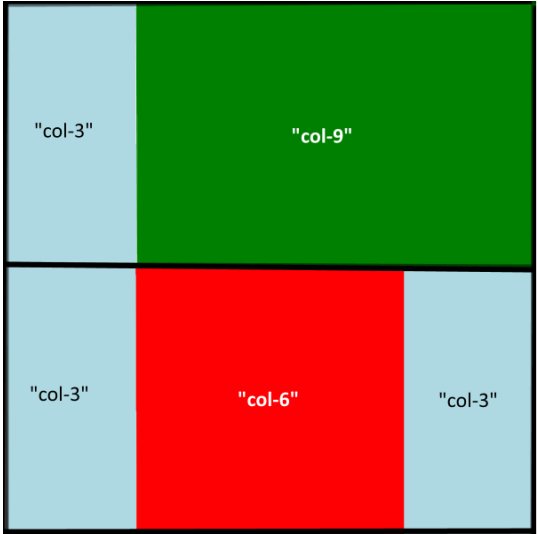
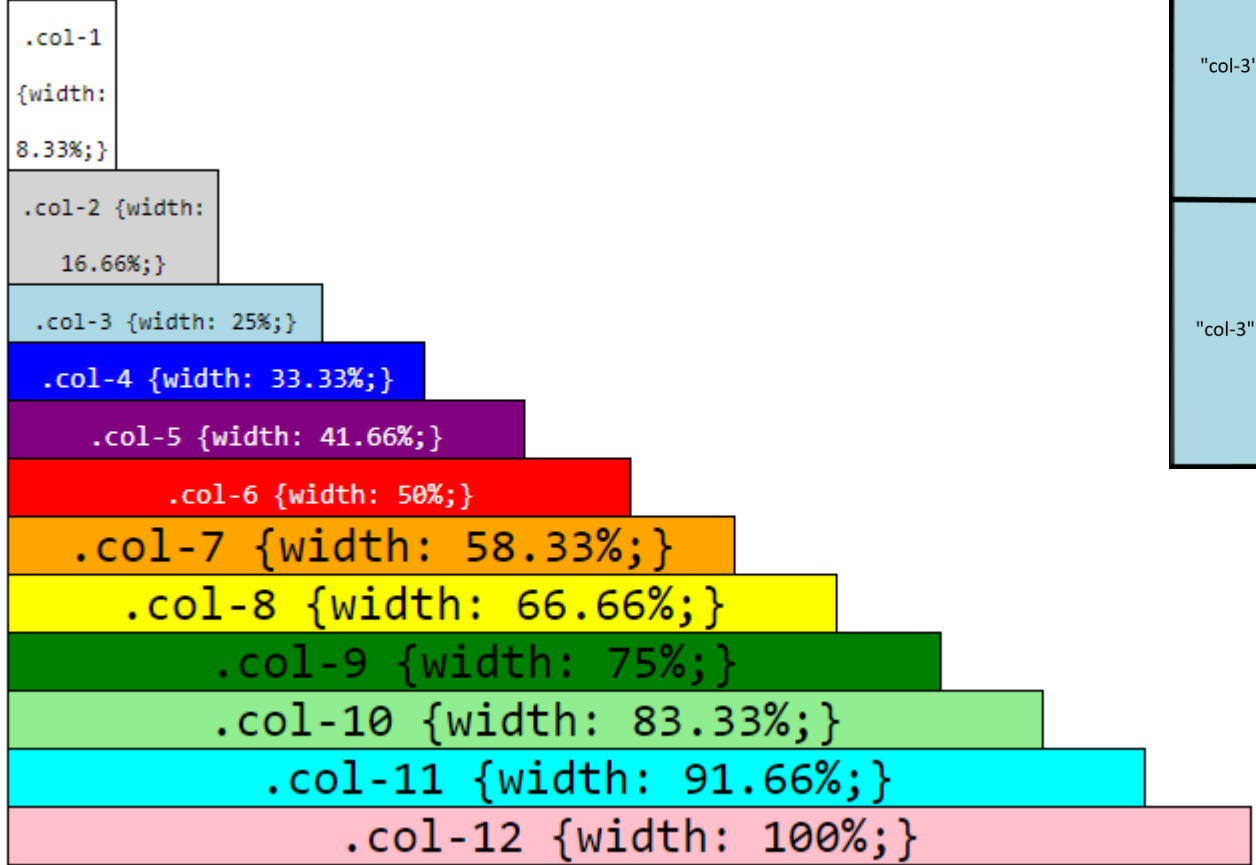
Организира се респонсив стил и `gutters [class*="col-"] { float: left; padding: 15px; }`

Изчислява се ширината на колона в %: $100\% / 12 \text{ columns} = 8.33\%$. След това се създава един клас за всяка от 12-те колони, `class="col-"` и число, определящо колко колони трябва да обхваща секцията.

Всеки ред е обвит в `<div>`. Редовете могат да се делят на различен брой колони, но броят на колоните в един ред винаги трябва да се събира до 12.



1	2	3	4	5	6	7	8	9	10	11	12



ред 1

ред 2

- Когато размерът на прозореца на браузъра стигне до много малка ширина, визуализацията не е добра. За това се използват `media queries` с точки на прекъсване и друг изглед за различните размери:

```
@media only screen and (max-width: 599px) {  
    /* For mobile phones: */ малки размери устройства  
    [class*="col-"] {  
        width: 100%; }  
}
```

- ▶ `/* Tablets: */ - среден размер устройства`
- ▶ `@media only screen and (min-width: 600px) {`
- ▶ `.col-t-1 {width: 8.33%;}`
- ▶ `.col-t-2 {width: 16.66%;}`
- ▶ `.col-t-3 {width: 25%;}`
- ▶ `.col-t-4 {width: 33.33%;}`
- ▶ `.col-t-5 {width: 41.66%;}`
- ▶ `.col-t-6 {width: 50%;}`
- ▶ `.col-t-7 {width: 58.33%;}`
- ▶ `.col-t-8 {width: 66.66%;}`
- ▶ `.col-t-9 {width: 75%;}`
- ▶ `.col-t-10 {width: 83.33%;}`
- ▶ `.col-t-11 {width: 91.66%;}`
- ▶ `.col-t-12 {width: 100%;}`
- ▶ `}`
- ▶ `<div class="col-3 col-t-3"></div>`

- Примери за сайтове с Responsive Grid System

<http://www.responsivegridsystem.com/>