

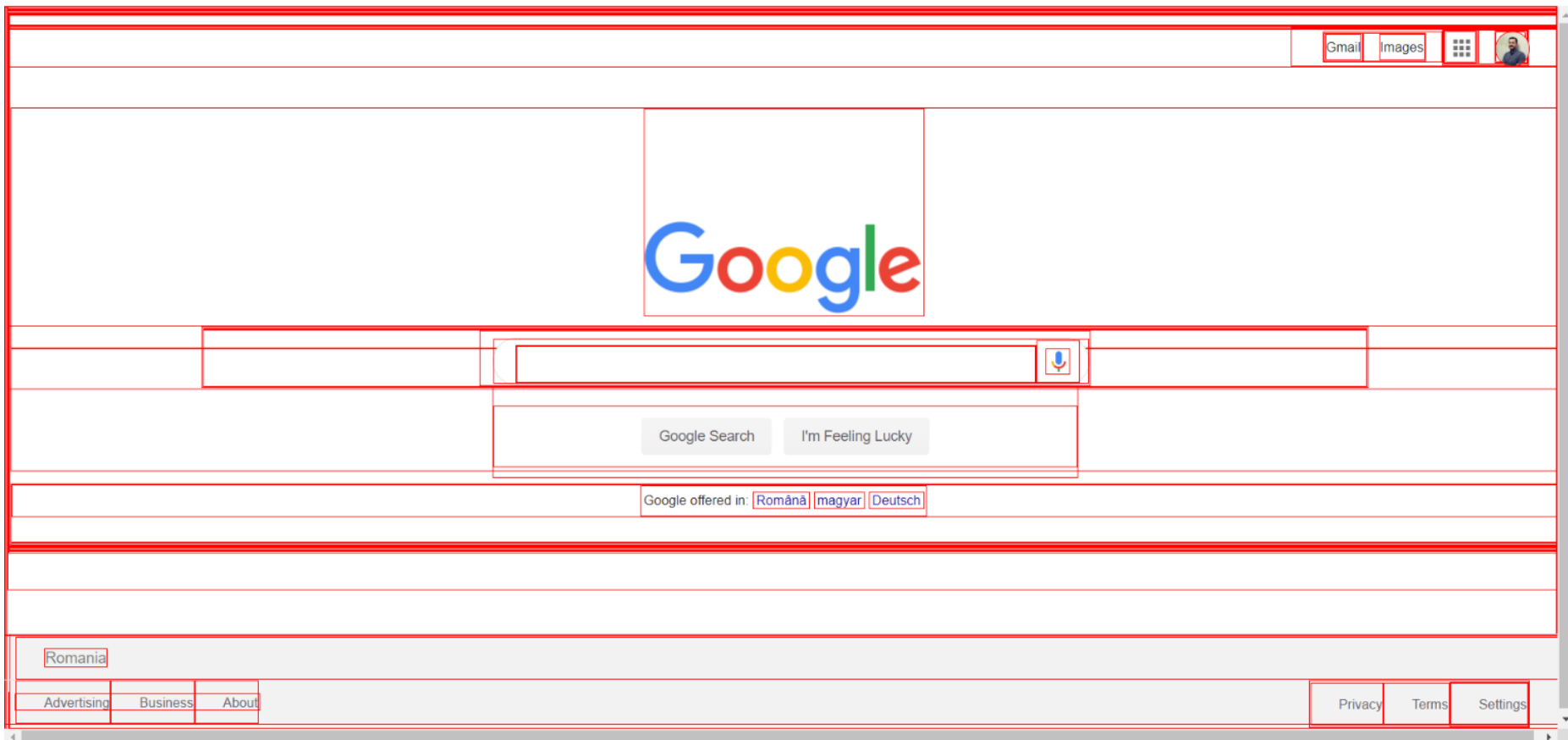


Saptamana 3

Partea 2

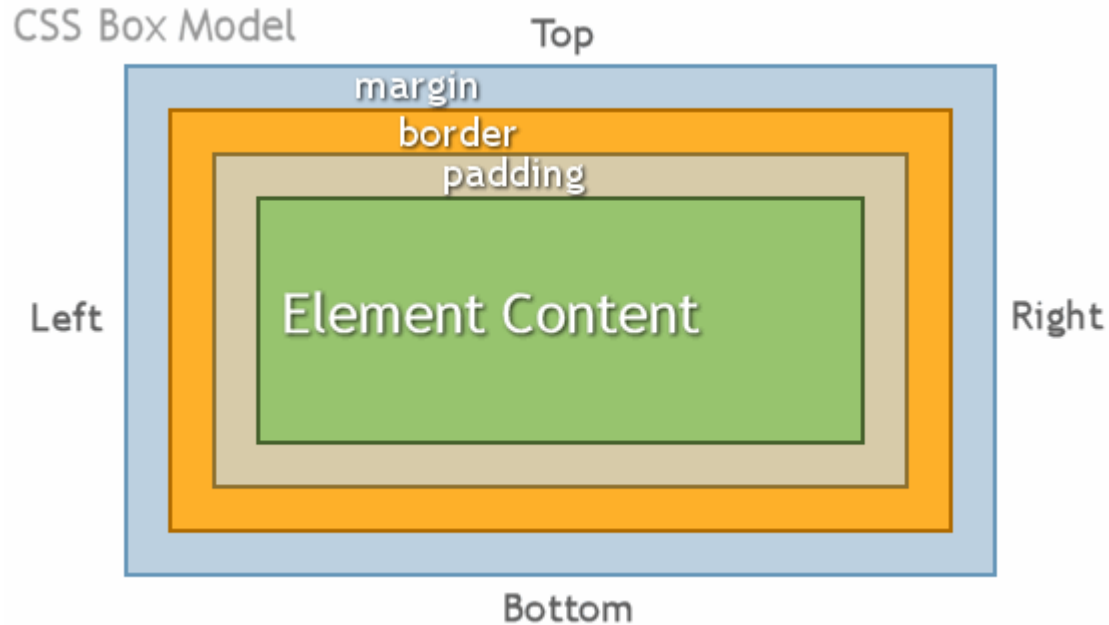
## Programare Front-End

# 1. Box Model



*Exemplu de pagina web si felul in care sunt delimitate elementele*

# Box Model



# Box Model

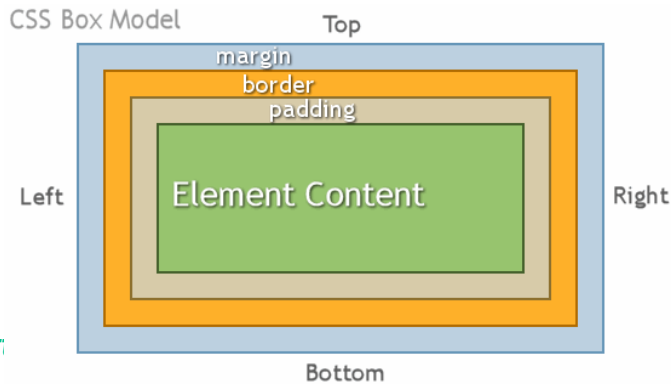
- Fiecare element din cadrul structurii HTML corespunzatoare unei pagini web are asociat un **container** sau o **cutie(box)**. Acest *container* este constituit din 4 zone foarte importante: **content, padding, border, margin**
- **Content** - este ceea ce se afla in **box** ca si continut (nucleu): o imagine sau un text de exemplu
- **Padding** - este zona care incapsuleaza continutul si ofera spatierea dintre continut si bordura
- **Border** - incapsuleaza *content + padding*
- **Margin** - incapsuleaza toate celelalte trei zone si este cea care delimiteaza spatiul elementului HTML relativ la celelalte elemente din jurul sau

# BOX MODEL – Padding

- **Padding**-ul unui element nu are culoare proprie dar va *lua culoarea* **content**-ului daca acesta are *background-color*
- Dupa cum se observa, exista 4 proprietati de tip padding:
  - *padding-left*
  - *padding-top*
  - *padding-right*
  - *padding-bottom*
- Toate cele 4 valori pot fi specificate in cadrul aceleiasi proprietati:

*padding: top right bot*

<https://codesandbox.io/s/lrz49r7oym>

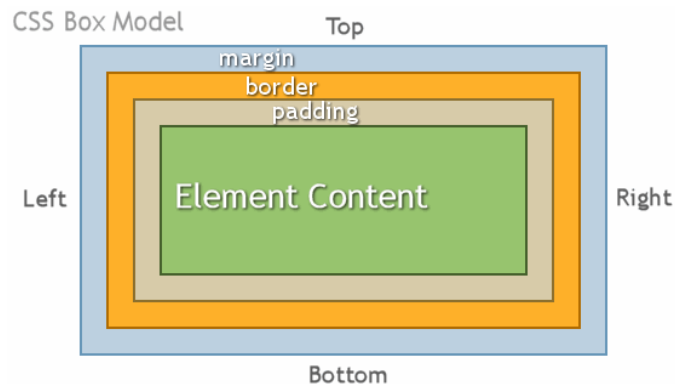


# BOX MODEL – Border

- **Border**, sau bordura unui element, incapsuleaza *padding + content*
- O bordura este definita de mai multe caracteristici:
  - *border-style (solid, dotted etc)*
  - *border-width*
  - *border-color*
  - *border-radius*
- Fiecare dintre aceste caracteristici sunt valabile pentru fiecare tip de bordura in parte: *top, right, bottom, left*.

Exemplu: *border-style-top, border-color-bottom*

<https://codesandbox.io/s/lrz49r7oym>



# BOX MODEL – Margin

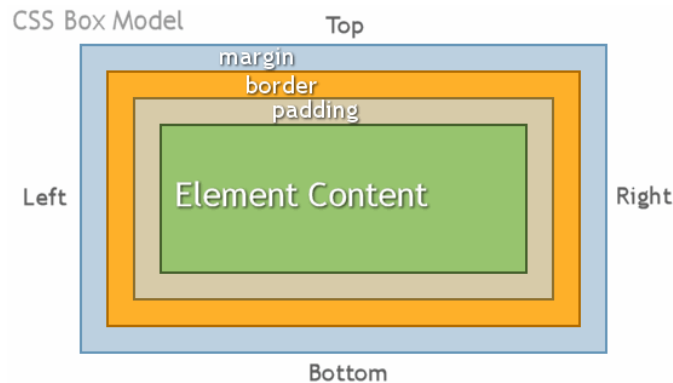
- **Margin** incapsuleaza *border + padding + content*
- Dupa cum vedem in poza putem avea 4 proprietati de margin:

- *margin-left*
- *margin-top*
- *margin-right*
- *margin-bottom*

- Toate cele 4 valori pot fi specificate in cadrul aceleiasi proprietati:

*margin: top right bottom left;*

<https://codesandbox.io/s/lrz49r7oym>





# Box Model – Content: Width & Height

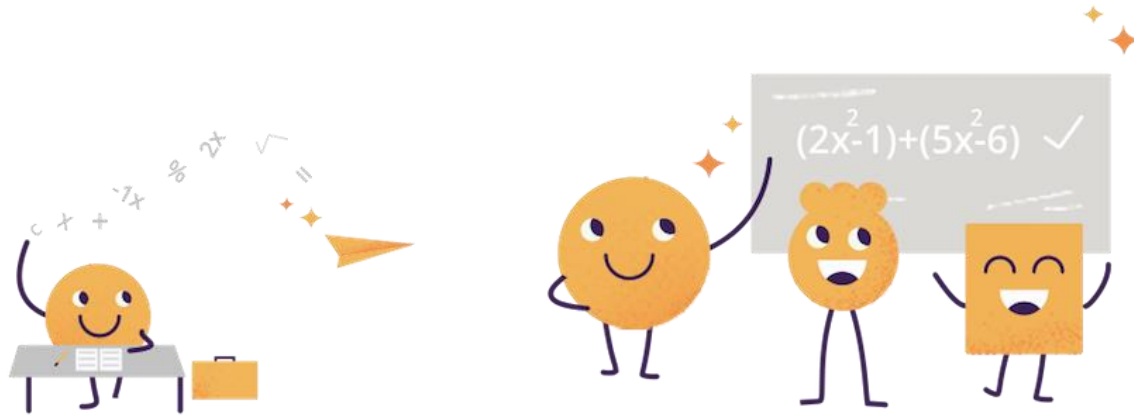
- **content**-ul unui element este definit ca si spatiu ocupat intr-un pagina de catre proprietatile **width** si **height**
- Cele 2 valori pot fi exprimate in *pixeli, procentaj*, etc.
- Valoarea de baza va fi **auto** - *content*-ul isi va ajusta dimensiunile in functie de ce contine (ex. in link-ul de mai jos)
- Daca dimenisunile setate vor fi prea mari iar continutul nu incapa, acesta va iesi inafara *box*-ului - ceea ce se numeste **overflow**
- **Overflow**-ul are 4 valori in plus fata de ce-a initiala (initial): *inherit, auto, scroll, hidden, visible*
  - [vezi exemple pentru tipuri de overflow](#)
- Pe langa proprietatile **width** si **height**, mai exista si **min-width, min-height** si **max-width, max-height**:  
<https://codesandbox.io/s/lrz49r7oym>

# BOX MODEL: Box-sizing

- Pentru ca **box-model**-ul era neintituv la inceput, s-a adaugat o proprietate noua numita **box-sizing** prin intermediul careia se poate specifica felul in care proprietile **width** si **height** ale unui element sunt calculate ( se iau sau nu in **padding** si **border** )
- Valoarea default este **box-sizing: content-box** - valorile proprietatilor *width* si *height* se vor aplica doar pe **content** (marimea *padding*-ului si a *border*-ului nu va fi luata in considerare)
- Daca dorim ca elementul sa pastreze mereu valorile specificate pentru **width** si **height**, vom folosi proprietatea **box-sizing** cu valoarea **border-box** - valorile vor fi aplicate pe intreg **box**-ul ( marimea *padding*-ului si a *border*-ului va fi luata in considerare )

Exemplu: <https://codesandbox.io/s/lrz49r7oym>

## PRACTICE: **BOX MODEL**

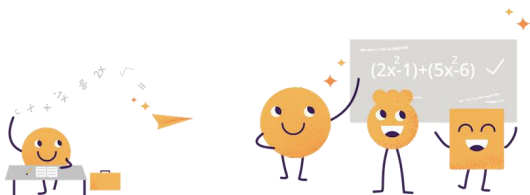


# PRACTICE: Box Model

## Cerinte:

*Creati o pagina care sa aiba urmatoarele caracteristici:*

- H1-urile sa aiba bordura rosie cu dimensiunea de 2px; Culoarea background-ului sa fie #6A6; Spatiul dintre continut si bordura sa fie de 10px (padding)*
- Sa contina o lista care sa aiba 15px spatiu in jurul ei (margin) si background-ul de culoarea #DE1E7E*
- Sa contina mai multe paragrafe, fiecare separat de cate un box de 500px pe 300px cu bordura de 2px neagra de tip dotted, iar culoarea background-ului sa fie #FB1*
- Sa contina mai multe link-uri cu bordura jos de tip dashed, albastra, iar culoarea background-ului acestora sa fie #a1a1a1.*



# PRACTICE: Box Model

## Cerinte:

1. Validati html-ul pe [https://validator.w3.org/#validate\\_by\\_input](https://validator.w3.org/#validate_by_input) si css-ul pe [https://jigsaw.w3.org/css-validator/#validate\\_by\\_input](https://jigsaw.w3.org/css-validator/#validate_by_input)



## 2. Display

# Display property

**Display** este cea mai importanta proprietate care ofera control asupra *layout*-ului; cu ajutorul *display* putem decide afisarea si modul acesteia pentru anumite elemente.

Cele mai des intalnite tipuri (valori) de *display* sunt urmatoarele:

- **none**
- **block**
- **inline**
- **inline-block**

**Atentie:** Pe langa “**display: none**” mai exista si cazul “**visibility: hidden**”, al carei efect pare sa aiba un comportament asemanator. Diferenta principala este insa faptul ca cea de-a doua in cauza pastreaza elementul in DOM si exista situatii in care acest lucru ar putea fi util

# Display property – Valori de baza

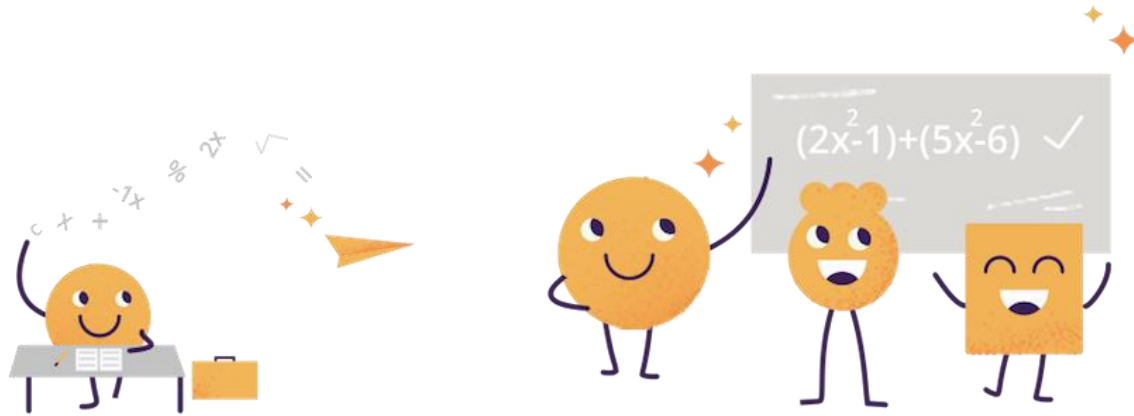
- Tipuri de elemente ce au display block by default: **headings, divs, paragraphs, forms, headers, footers, sections**
- Tipuri de elemente ce au display inline by default: **spans, anchors (< a href..), imgs**

<https://codesandbox.io/s/lrz49r7oym>



# PRACTICE: Display types

<https://codesandbox.io/s/0pz4zym80v>



### 3. Positioning of elements

# Positioning

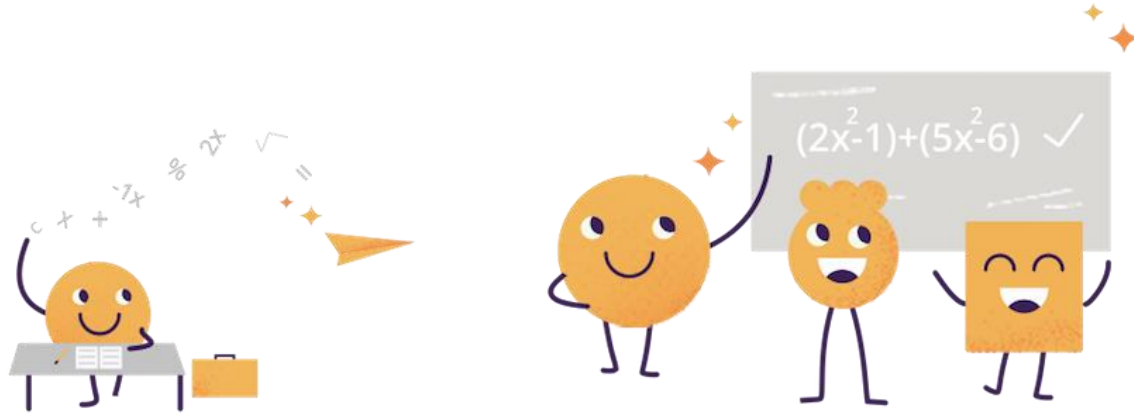
- Pentru a crea *layout*-uri complexe avem sa putem controla modul in care sunt pozitionate anumite elemente relativ la celelalte; proprietatea **position** ne vine in ajutor in acest sens
- Exista 4 valori pentru aceasta proprietate
  - **static** - este pozitia de baza a oricarui element si este deja setata atunci cand un element este creat
  - **relative** - aditional pozitiei statice, pozitia relativa ofera posibilitatea specificarii de proprietati aditionale pentru un anumit element (*top, left, right, bottom*)
  - **fixed** - elementul devine fix, pozitionat relativ fata de **viewport**. Va fi mereu 'lipit' de fereastra, ceea ce inseamna ca va ramane in aceeasi pozitie indiferent daca vom face *scroll*

# Positioning

- **absolute** - se comporta asemanator cu *fixed* doar ca este elementul este pozitionat relativ fata de primul element ce are pozitie relativa. Daca nu exista niciun element cu pozitie relativa, se va pozitiona relativ fata de *body*

<https://codesandbox.io/s/r4wy0wm1jm>

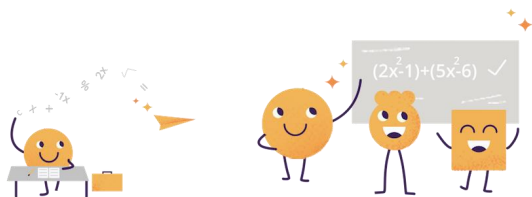
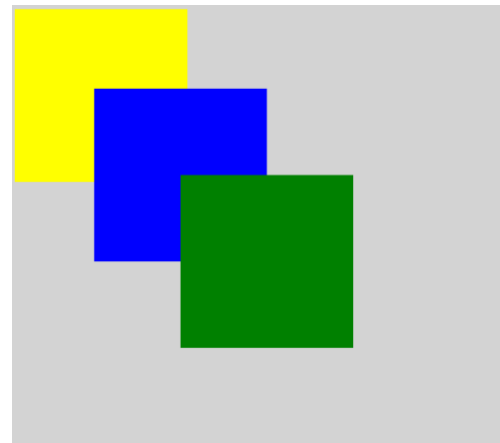
## PRACTICE: Positioning



# PRACTICE: Positioning

## Cerinte:

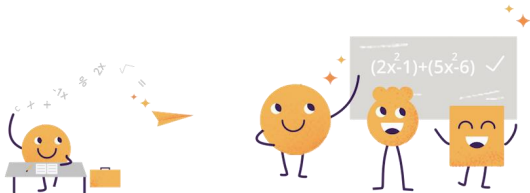
1. *Recreati imaginea de alaturi*
1. *Bonus points: pozitionati patratul galben in mijlocul paginii - celelalte patrate trebuie sa ramana in continuare pozitionate la fel fata de acesta*



# PRACTICE: Positioning

**Rezolvare:**

<https://codesandbox.io/s/q10npj1jkw>



# PRACTICE: Positioning

## Cerinte:

### 1. *Recreati imaginea de mai jos folosind proprietatea position:*

<nav>

- Home
- Taco Menu
- Draft List
- Hours
- Directions
- Contact

</nav>

<section>

The `margin-left` style for `sections` makes sure there is room for the `nav`. Otherwise the absolute and static elements would overlap

</section>

<section>

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Phasellus imperdiet, nulla et dictum interdum, nisi lorem egestas odio, vitae scelerisque enim ligula venenatis dolor. Maecenas nisl est, ultrices nec congue eget, auctor vitae massa. Fusce luctus vestibulum augue ut aliquet. Mauris ante ligula, facilisis sed ornare eu, lobortis in odio. Praesent convallis urna a lacus interdum ut hendrerit risus congue. Nunc sagittis dictum nisi, sed ullamcorper ipsum dignissim ac. In at libero sed nunc venenatis imperdiet sed ornare turpis. Donec vitae dui eget tellus gravida venenatis. Integer fringilla congue eros non fermentum. Sed dapibus pulvinar nibh tempor porta. Cras ac leo purus. Mauris quis diam velit.

</section>

<section>


Notice what happens when you resize your browser. It works nicely!

</section>

</footer>

If you use a fixed header or footer, make sure there is room for it! I put a `margin-bottom` on the `body`.

</footer>

 **wantsome**  
The friendly IT Academy

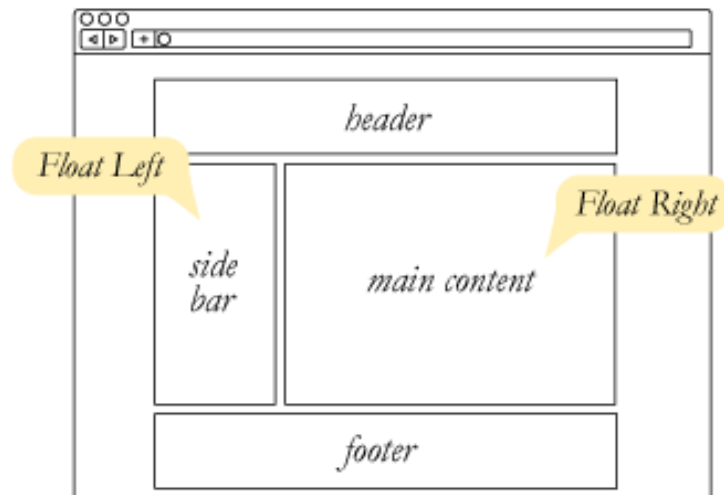


## 4. Floating of elements

# Float

- O alta proprietate care nu mai este atat de des folosita si poate ajuta la aranjarea elementelor intr-o pagina, este **float**

<https://codesandbox.io/s/10vv5lz5ll>



# PRACTICE: Floating

## Cerinte:

### 1. *Recreati imaginea de mai jos folosind proprietatea float:*

<nav>

- Home
- Taco Menu
- Draft List
- Hours
- Directions
- Contact

</nav>

<section>

The `margin-left` style for `sections` makes sure there is room for the `nav`. Otherwise the absolute and static elements would overlap

</section>

<section>

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Phasellus imperdiet, nulla et dictum interdum, nisi lorem egestas odio, vitae scelerisque enim ligula venenatis dolor. Maecenas nisl est, ultrices nec congue eget, auctor vitae massa. Fusce luctus vestibulum augue ut aliquet. Mauris ante ligula, facilisis sed ornare eu, lobortis in odio. Praesent convallis urna a lacus interdum ut hendrerit risus congue. Nunc sagittis dictum nisi, sed ullamcorper ipsum dignissim ac. In at libero sed nunc venenatis imperdiet sed ornare turpis. Donec vitae dui eget tellus gravida venenatis. Integer fringilla congue eros non fermentum. Sed dapibus pulvinar nibh tempor porta. Cras ac leo purus. Mauris quis diam velit.

</section>

<section>


Notice what happens when you resize your browser. It works nicely!

</section>

</footer>

If you use a fixed header or footer, make sure there is room for it! I put a `margin-bottom` on the `body`.

</footer>

 **wantsome**  
The friendly IT Academy

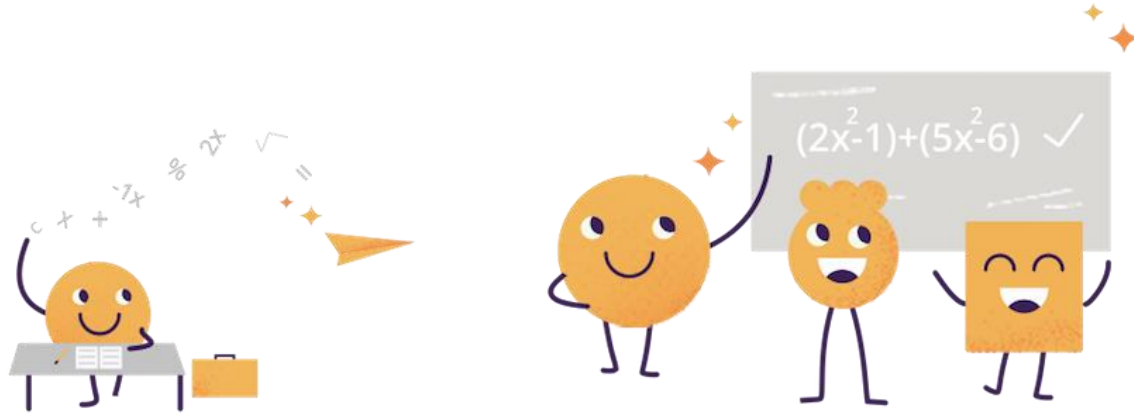
# Backgrounds

- Un background al unui element este definit de urmatoarele caracteristici :
  - **background-color**
  - **background-image** (aici avem `url()`, `linear-gradient()`, `radial-gradient()`, etc...)
  - **background-position**
  - **background-size**
  - **background-repeat**
  - **background-clip**
  - **background-origin**
  - **background-attachment**

Exemplu: **background: lightblue url("img\_tree.gif") no-repeat fixed center;**

**background: bg-color bg-image position/bg-size bg-repeat bg-origin bg-clip bg-attachment initial|inherit;**

## PRACTICE: **Background**



# PRACTICE: Background

## Cerinte:

1. *Recreati imaginea de mai jos folosind proprietatea background (hint: **clip**):*

