TEHNICI WEB

CSS #2

Claudia Chiriță . 2024/2025

PSEUDO-CLASE

PSEUDO-CLASE

```
selector:pseudo-class {
  proprietate: valoare;
}
```

- cuvânt-cheie adăugat unui selector ce specifică o stare specială a elementului selectat
- ex. folosite pentru definirea stilului
  - unui element la trecerea cu mouse-ul peste el
  - diferit pentru legături vizitate și nevizitate

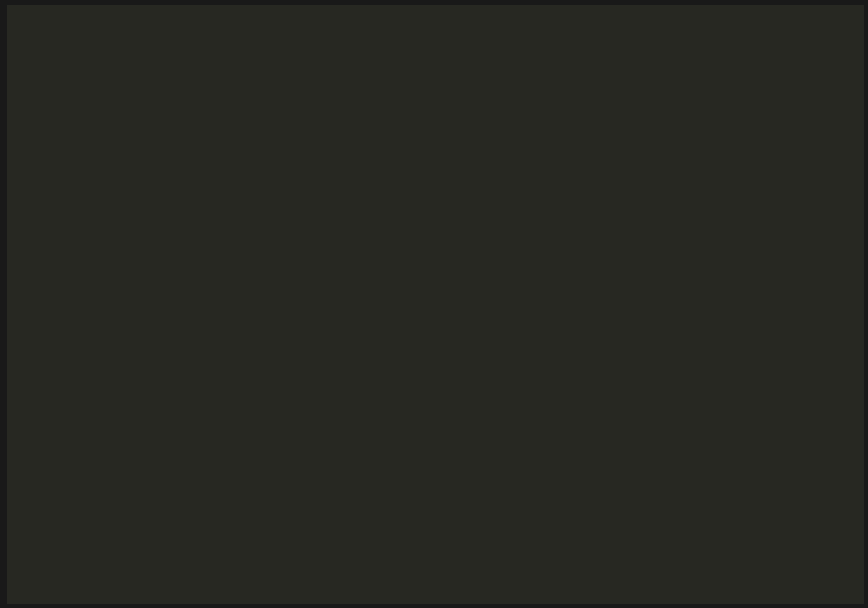
PSEUDO-CLASE

```
:link /* legături care nu au fost vizitate */
:visited /* legături care au fost vizitate */
:hover  /* elemente peste care se trece cu mouse-ul */
:active /* legături active */
:not() /* elementele care nu apar în argumentul lui not */
```

PSEUDO-CLASA :LINK

```
<!DOCTYPE html>
<html lang="en">
  <head>
   <style type="text/css">
     a {color:white;}
     a:link {color:red;
             text-decoration:none;}
     a:visited {color:cyan;
                text-decoration:none;}
   </style>
  </head>
  <body>
   <a href="https://github.com">visited</a>
<br>
   <a
```

›



PSEUDO-CLASA :HOVER

```
<!DOCTYPE html>
<html lang="en">
  <head>
   <style type="text/css">
      p {color:white;}
      p:hover {color:red;
               text-decoration:underline;}
      a {color:white;}
      a:active {background-color:yellow;
color:black;}
   </style>
  </head>
  <body>
    <p>hover over me</p>
    <a>click me</a>
  </body>
</html>
```

›

PSEUDO-CLASA :NOT

```
<!DOCTYPE html>
<html lang="ro">
  <head>
   <style type="text/css">
      p {color:white;}
      p:not(.foo){color:red;}
   </style>
  </head>
  <body>
    <p>paragraf fără clasă</p>
    <p class="foo">paragraf cu .foo</p>
  </body>
</html>
```

PSEUDO-CLASA :FIRST-CHILD

selectează un element care este primul copil al unui alt element

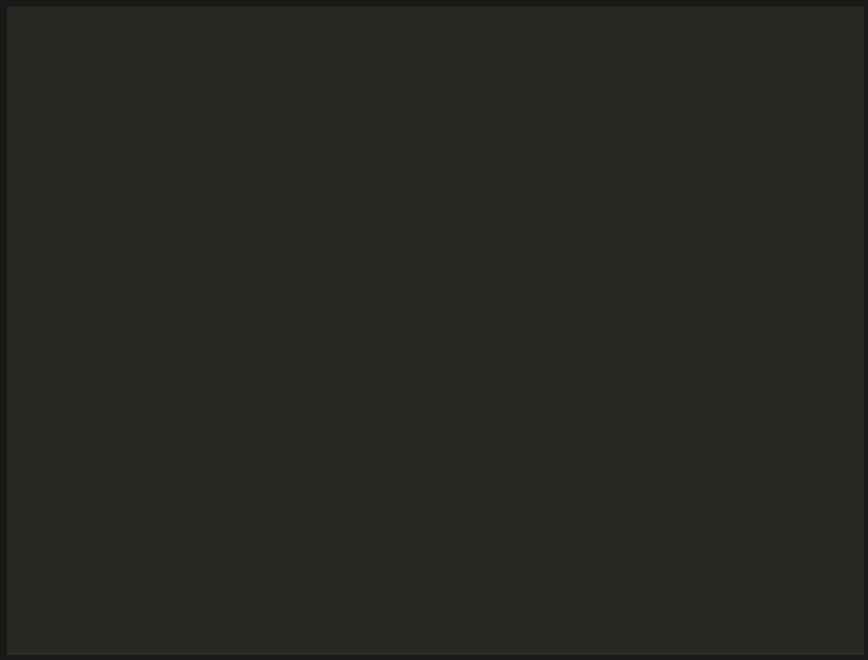
```
<!DOCTYPE html>
<html lang="ro">
  <head>
   <style type="text/css">
      p {color:white;}
      p:first-child {color:red;}
      p:last-child {color:cyan;}
   </style>
  </head>
  <body>
    <p>paragraf 1</p>
    <p>paragraf 2</p>
    <div><p>paragraf 3</p>
    <p>paragraf 4</p></div>
  </body>
</html>
```

›

PSEUDO-CLASA :FIRST-CHILD

```
<!DOCTYPE html>
<html lang="en">
  <head>
   <style type="text/css">
      p {color:white;}
      p:first-child b{color:red;}
      /* p b:first-child {color:red;} */
   </style>
  </head>
  <body>
    <p>the panda <b>eats</b>, <b>shoots</b>
& <b>leaves</b></p>
    <p>the panda eats <b>shoots</b> &
<b>leaves</b></p>
  </body>
</html>
```

›



PSEUDO-CLASE

`:first-of-type`
selectează elementul care este primul copil de tipul specificat al unui alt element

`:nth-of-type(n)`
selectează elementul care este al n-lea copil de tipul specificat al unui alt element

`:nth-child(n)`
selectează elementul de tipul specificat care este al n-lea copil al unui alt element

n poate fi număr natural, cuvânt cheie (*odd*, *even*), formulă (an+b, cu n pornind de la 0)

PSEUDO-CLASE

```
<!DOCTYPE html>
<html lang="ro">
  <head>
   <style type="text/css">
      p {color:white;}
      p:nth-of-type(2){color:red;}
      p:nth-of-type(odd){color: cyan;}
      p:nth-of-type(4n+0){color: pink;}
   </style>
  </head>
  <body>
    <p>paragraf 1</p>
    <p>paragraf 2</p>
    <p>paragraf 3</p>
    <p>paragraf 4</p>
    <p>paragraf 5</p>
    <p>paragraf 6</p>
    <p>paragraf 7</p>
    <p>paragraf 8</p>
  </body>
</html>
```

›

PSEUDO-CLASE

```
<!DOCTYPE html>
<html lang="en">
  <head>
   <style type="text/css">
      * {color:white;}
      p:nth-of-type(3){color:red;}
      p:nth-child(3){color:cyan;}
   </style>
  </head>
  <body>
    <div>div</div>
    <p>paragraf 1</p>
    <p>paragraf 2</p>
    <p>paragraf 3</p>
    <p>paragraf 4</p>
    <p>paragraf 5</p>
    <p>paragraf 6</p>
    <p>paragraf 7</p>
    <p>paragraf 8</p>
  </body>
</html>
```

›

PSEUDO-ELEMENTE

PSEUDO-ELEMENTE (::)

- folosite pentru a defini stilul unei părți a unui element
- elemente virtuale, create cu CSS, care nu apar în arborele HTML al paginii

```
::first-letter /* prima literă a unui text */
::first-line /* prima linie a unui element block */

::before, ::after /* folosite pentru a adăuga conținut*/
                  /* înainte/după un element HTML; */
                  /* folosite cu proprietatea content */
```

PSEUDO-ELEMENTE (::)

```
<!DOCTYPE html>
<html lang="en">
  <head>
   <style type="text/css">
      * {color:white;}
      p::first-letter {font-size: 1.5rem;
                       font-weight: bold;
                       color: red;}
   </style>
  </head>
  <body>
    <p>Flatland: A Romance of Many
Dimensions is a satirical novella by the
English schoolmaster Edwin Abbott Abbott,
first published in 1884. The story describes
a two-dimensional world inhabited by
geometric figures; women are line segments,
while men are polygons with various numbers
of sides. </p>
    <a href="https://www.youtube.com/watch?
v=yBbZmwROv84">animation</a>
```

›

VARIABILE CSS

- proprietăți CSS custom (nestandard, definite de utilizator)
- definite pentru a fi reutilizate în cadrul unui document

VARIABILE CSS

- globale: valabile în întreg documentul

```
:root {
--nume-proprietate: valoare;}
```

- locale: valabile în interiorul (toți descendenții) elementului în care au fost definite

```
element {
--nume-proprietate: valoare;}
```

- se folosesc cu funcția

```
var(--nume-proprietate)
```

CSS LAYOUT

PROPRIETATEA POSITION

- tipul de poziționare a unui element în pagină
- poate avea valorile:

```
position:static;
position:relative;
position:absolute;
position:fixed;
position:sticky;
```

- pentru poziționare se folosesc proprietățile *left*, *right*, *top* și *bottom*

POSITION:STATIC

- poziția implicită a oricărui element HTML
- cu poziția *static* elementul va avea un flux normal în pagină
- elementele cu poziția *static* nu sunt afectate de proprietățile left, right, top, bottom

POSITION:RELATIVE

- un element cu *position:relative* este poziționat relativ față de poziția pe care ar fi avut-o în mod normal în pagină
- pentru poziționare se folosesc proprietățile *left*, *right*, *top* și *bottom*

POSITION:ABSOLUTE

- elementul este poziționat relativ față de primul părinte care are *position:absolute/relative/fixed*
- elementul este scos din fluxul documentului
- se pot suprapune cu alte elemente (exit Flatland!)

POSITION:FIXED

- elementul are o poziție fixă în fereastra de browser
- elementul este scos din fluxul documentului și nu e afectat de scroll
- pentru poziționare se folosesc proprietățile *left*, *right*, *top* și *bottom*

POSITION:STICKY

- elementul comută între poziționarea relativă și cea fixă în funcție de scroll
- dacă poziția scrollului nu depășește elementul, acesta va fi poziționat relativ, altfel se comportă ca un element fix

SUPRAPUNERI: Z-INDEX

```
z-index: ...,-100 /* 0, 100,... */
```

- prin poziționare, elementele se pot suprapune; pot apărea stive de elemente
- ordinea elementelor în stivă este dată de *z-index*
- elementele cele mai vizibile au *z-index* mai mare
- *z-index* poate fi setată doar pentru elementele care sunt poziționate *absolute*, *relative* sau *fixed*

PROPRIETATEA FLOAT

```
float: left; /* right, none */
```

- elementele cu proprietatea *float* sunt scoase din fluxul documentului și poziționate conform valorii, la stânga sau la dreapta
- ele afectează celelalte elemente: se aranjează înconjurând ("wrapping") elementele *float*
- elementele poziționate absolut ignoră prop. *float*

PROPRIETATEA CLEAR

```
clear: left; /* right, none, both */
```

- determină dacă un element trebuie poziționat sub elemente *float* care îl precedă
- elementele cu proprietatea *clear* nu se aranjează în jurul elementelor *float*, ci se poziționează sub acestea

FLEXBOX

- metodă de layout unidimensională pentru aranjarea itemurilor în rânduri și coloane
- itemurile se extind (*flex*) pentru a umple spațiul disponibil sau se micșorează pentru a încăpea în spații mai mici

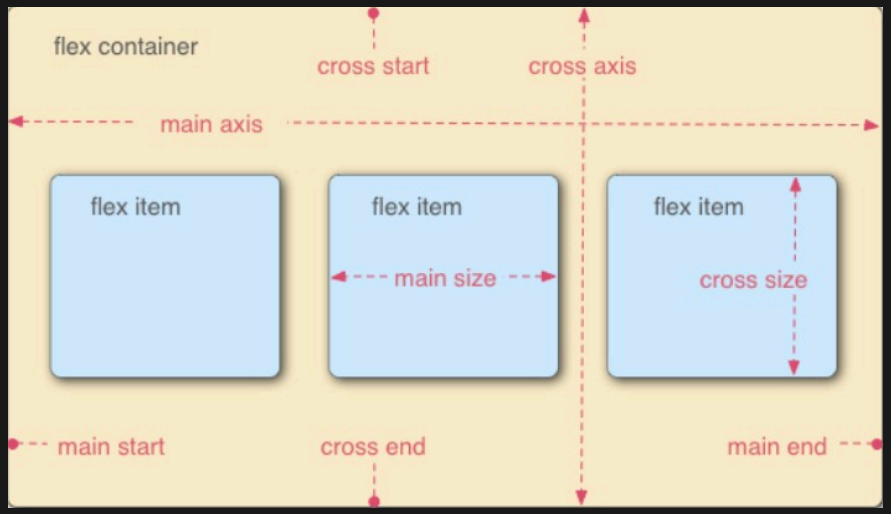
FLEXBOX

- layoutul flexibil oferă o metodă eficienta de a așeza, alinia și a distribui spațiul între elementele din document chiar și atunci când dimensiunea viewportului și a elementelor este necunoscută sau dinamică

tutorial flexbox

FLEXBOX

- layoutul flexibil presupune definirea a două tipuri de elemente:
  - **flex container** - element părinte
  - **flex items** - elemente copii



FLEX CONTAINER

proprietăți pentru părinte:

```
display
flex-direction
flex-wrap
flex-flow
justify-content
align-items
align-content
```

DISPLAY FLEX

```
display: flex; /* inline-flex */
```

- definește un container flexibil inline sau block, în funcție de valoarea dată
- creează un context flexibil pentru toți copiii săi direcți

FLEX DIRECTION

```
flex-direction: row; /* row-reverse, column, column-reverse*/
```

- stabilește direcția în care sunt plasate itemurile în container: pe orizontală (stânga-dreapta sau dreapta-stânga) sau pe verticală (sus-jos sau jos-sus)

FLEX WRAP

```
flex-wrap: wrap; /* nowrap, wrap-reverse*/
```

- stabilește dacă itemurile vor fi aranjate pe o singură linie (coloană) sau pe mai multe
- implicit, itemurile se vor aranja toate pe o linie (coloană)

FLEX FLOW

```
flex-flow: flex-direction flex-wrap;
```

- prescurtare pentru proprietățile flex-direction și flex-wrap
- implicit: flex-flow: row nowrap

JUSTIFY CONTENT

```
justify-content: start; /* end, center, space-between,
                           space-around, space-evenly */
```

- specifică modul în care sunt aliniate itemurile de-a lungul axei principale

ALIGN ITEMS

```
align-items: stretch; /* start, end, center, baseline */
```

- specifică modul în care sunt aliniate itemurile pe axa secundară (axa perpendiculară pe axa principală)

ALIGN CONTENT

```
align-content: stretch; /*  start, end, center,
                            space-between, space-around */
```

- distribuie spațiul dintre și în jurul itemurilor pe axa transversală
- nu are efect dacă există o singură linie în container

FLEX ITEMS

proprietăți pentru copii:

```
order
flex-grow
flex-shrink
flex-basis
flex
align-self
```

ORDER

```
order: <întreg>; /* implicit 0*/
```

- stabilește ordinea în care apar itemurile în containerul flexibil
- implicit, itemurile apar în ordinea în care sunt scrise în documentul HTML

FLEX GROW

```
flex-grow: <număr>; /* implicit 0*/
```

- stabilește capacitatea itemurilor de a-și mări dimensiunea dacă este spațiu disponibil în interiorul containerului
- dacă toate itemurile au flex-grow egal cu 1, ele vor împarți spațiul disponibil în mod egal

FLEX SHRINK

```
flex-shrink: <număr>; /* implicit 1*/
```

- definește capacitatea itemurilor de a se micșora dacă este necesar

FLEX BASIS

```
flex-basis: <length>; /* auto (implicit) */
```

- definește dimensiunea implicită a unui element înainte de distribuirea spațiului rămas

FLEX

```
flex: flex-grow flex-shrink flex-basis;
```

- prescurtare pentru proprietățile *flex-grow, flex-shrink* și *flex- basis*
- implicit flex: 0 1 auto

ALIGN SELF

```
align-self: stretch; /* start, end, center, baseline, auto */
```

- permite alinierea individuală a itemurilor pe axa secundară
- înlocuiește alinierea specificată cu *align-items*

WHY FLEX?

situații tipice în care folosim flexbox

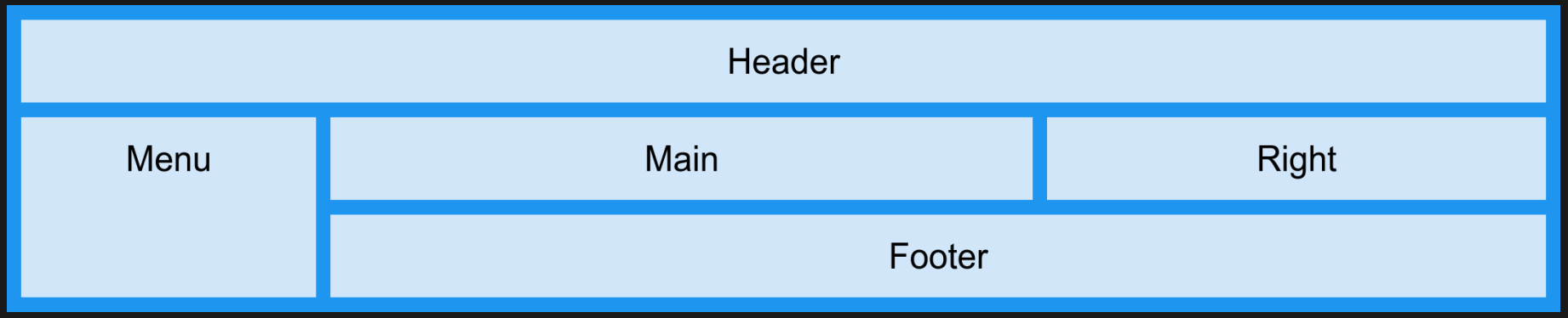
GRID LAYOUT

- sistem de layout bidimensional, bazat pe rețea, cu rânduri și coloane, ce ușurează modul de a așeza, alinia și a distribui spațiul între elementele din document

tutorial grid

GRID LAYOUT

- CSS grid presupune definirea a două tipuri de elemente:
  - **grid container** - element părinte
  - **grid items** - elemente copii



GRID CONTAINER

proprietăți pentru părinte:

```
display
grid-template-columns & grid-template-rows
grid-template-areas
column-gap & row-gap
justify-content
align-content
justify-items & align-items
```

DISPLAY GRID

```
display: grid; /* inline-grid */
```

- definește elementul ca un grid container inline sau block, în funcție de valoarea dată
- stabilește un context de formatare a grilei pentru toți copiii săi direcți

GRID TEMPLATE

`grid-template-columns`
`grid-template-rows`

- definește numărul de coloane și rânduri în care este împărțit gridul și lățimea/înălțimea acestora (unități de măsură: %, px, auto, fr)

`grid-template-areas`

GAPS

`column-gap`
`row-gap`

- definește spațiul dintre coloane și dintre rânduri

mind the gap?

JUSTIFY CONTENT

```
justify-content: start; /* end, center, space-between,
                           space-around, space-evenly */
```

- specifică modul în care este aliniată grila în interiorul containerului
- alinierea se face pe orizontală

ALIGN CONTENT

```
align-content: stretch; /*  start, end, center,
                           space-between, space-around, space-eve
```

- specifică modul în care este aliniată grila în interiorul containerului, pe verticală

```
place-content: align-content justify-content
```

JUSTIFY & ALIGN ITEMS

```
justify-items: stretch; /* start, end, center */
```

- specifică modul în care sunt aliniate itemurile pe axa principală

```
align-items: stretch; /* start, end, center, baseline */
```

- specifică modul în care sunt aliniate itemurile pe axa secundară

```
place-items: align-items justify-items
```

GRID ITEMS

proprietăți pentru copii:

```
grid-column & grid-row
grid-area
justify-self & align-self
place-self
```

GRID COLUMN & ROW

```
grid-column: start-coloană / end-coloană
```

- definește coloanele în care se plasează un element
- prescurtare pentru proprietățile grid-column-start și grid-column-end

```
grid-row: start-linie / end-linie
```

- definește rândurile în care se plasează un element
- prescurtare pentru proprietățile grid-row-start și grid-row-end

GRID AREA

```
grid-area: start-linie / start-coloană
           / end-linie / end-coloană
```

- prescurtare pentru proprietățile grid-row-start, grid-column-start, grid-row-end, grid-column-end

GRID AREA

```
grid-area: nume-item
```

- poate denumi un item al gridului care este referit apoi folosind proprietatea grid-template-areas pentru container

JUSTIFY & ALIGN SELF

```
justify-self: stretch /* start, end, center */
```

- aliniază un item al gridului în interiorul celulei, pe axa principală

```
align-self: stretch /* start, end, center */
```

- aliniază un item al gridului în interiorul celulei, pe axa secundară

```
place-self: align-self justify-self
```

FLEXBOX VS. GRID


(\ /)
(. .)
C(")(")
întrebări?