FLEXBOX

FLEXBOX

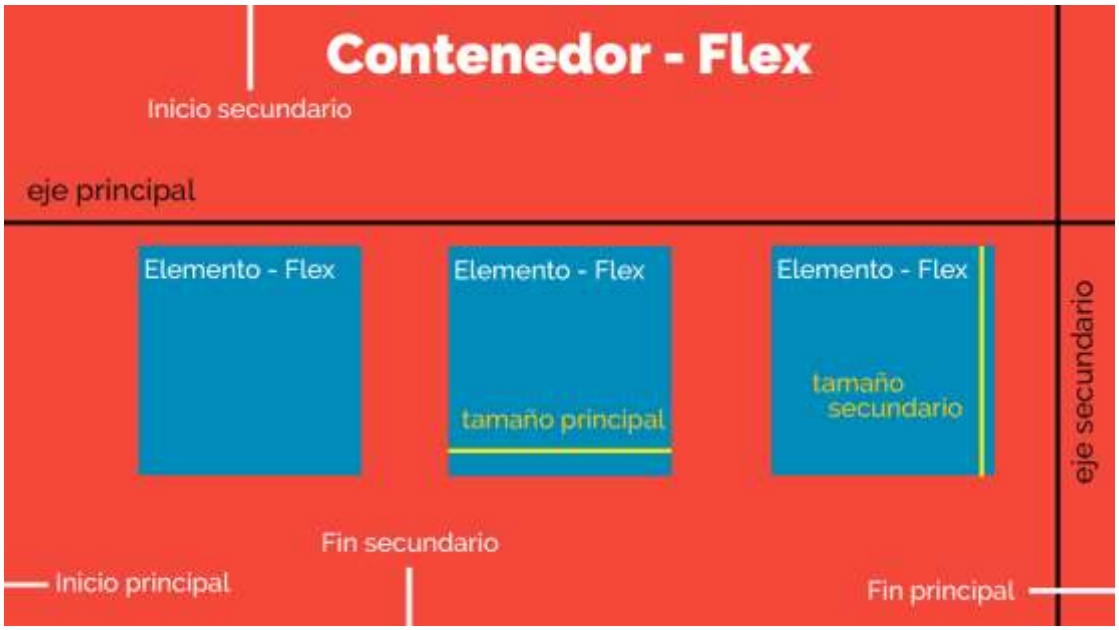
Flexbox és un nou mòdul de disseny en CSS3 per millorar la forma en què fem **Responsive Design** , evitant així l'ús de **float** *(No facis servir float per fer responsive design).*

Hola Flexbox, adeu float!

FlexBox facilita la manera de posicionar elements, és més simple i fa servir menys codi.

Per començar a treballar amb FlexBox primer has d'entendre l'estructura, FlexBox està constituït per:

- un pare **(Contenidor - Flex)** i
- els seus fills **(Element - Flex).**



- **Contenidor-Flex:** És l'element "**pare**" que conté els elements "fills", per definir-lo s'usa "**flex**" o "**inline-flex**" a la propietat **display** .
- **Element-Flex:** Aquests són els elements "fills" que tindran un comportament automàtic depenent el que defineixi l'element "pare".
- **Eixos:** Cada disseny "FlexBox" està compost per dos eixos.

  Eix principal: És l'eix que defineix la posició horitzontal dels **elements - Flex**

  Eix secundari: És l'eix que defineix la posició vertical dels **elements - Flex.**
- **Direccions:** Les parts **d'inici principal/fi principal** i **inici secundari/fi secundari** del **contenidor-Flex** defineixen l'origen i final del flux **d'elements - Flex.**
- **Dimensions:** Això equival a l'amplada **(mida principal)** i l'alçada **(mida secundari)** de **l'element - Flex** que depenen **d'eix principal** i **eix secundari.**

Actualment Flexbox és compatible amb tots els navegadors.

DEFININT FLEXBOX

Per començar a fer servir FlexBox només hem de definir `flex` a la propietat `display` de l'element pare:

```
<section class="pare"> </section>
```

```
.pare{ display: flex; }
```

Nota: Això és l'únic que necessitem per configurar el contenidor principal i automàticament tots els seus fills es convertiran en elements **flex.**

PROPIETAT DEL CONTENIDOR - FLEX (PARE)

El **contenidor - flex (pare)** té una sèrie de propietats interessants:

FLEX-DIRECTION

Valor por defecte: row

Aquesta propietat defineix les adreces de l'**eix principal**, és a dir, on es mouran els **Elements - Flex**, tant horitzontalment com verticalment.

Horitzontalment:

```
.pare {
  flex-direction: row;
}
```

Els elements **- flex** s'apilen en una fila d'esquerra a dreta definint FlexBox

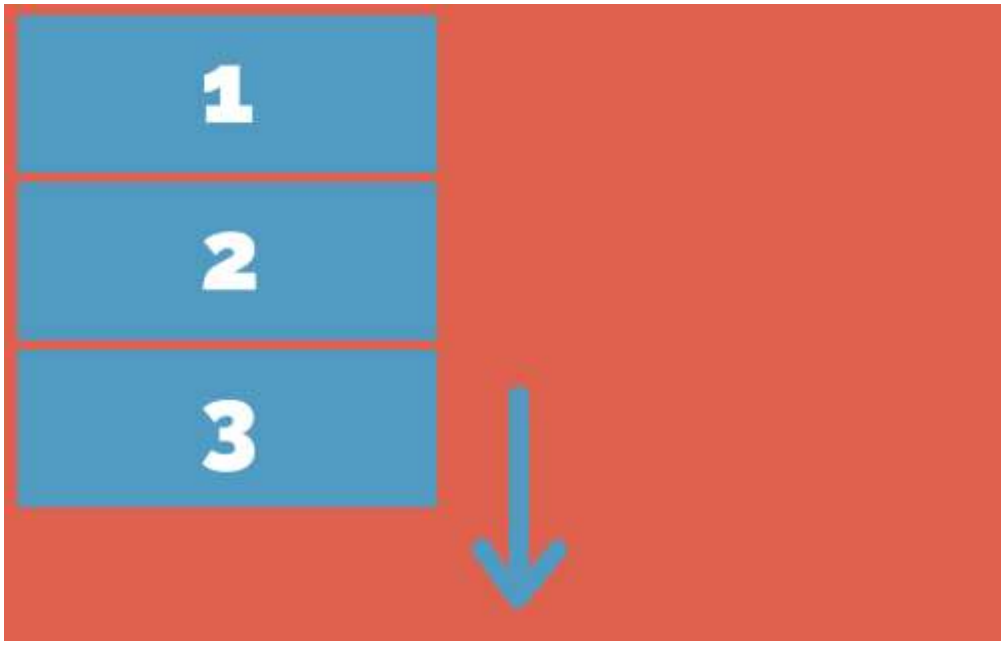

```
.pare { flex-direction: row-reverse; }
```

Els **elements - flex** s'apilen en una fila de dreta a esquerra.

verticalment:

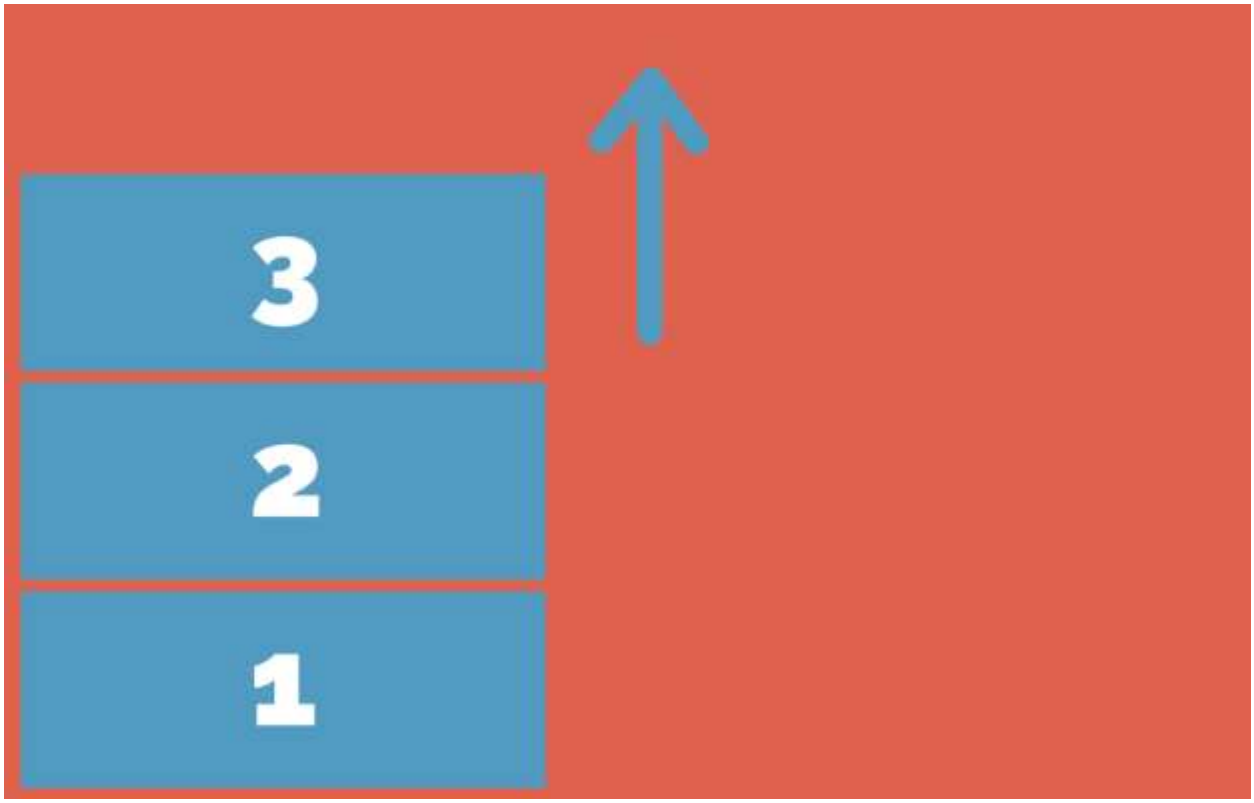
```
.pare { flex-direction: column; }
```

Els **elements - flex** s'apilen en una columna de dalt a baix.




```
.pare { flex-direction: column-reverse; }
```

Els **elements - flex** s'apilen en una columna de baix a dalt.



FLEX-WRAP

Valor per defecte: nowrap

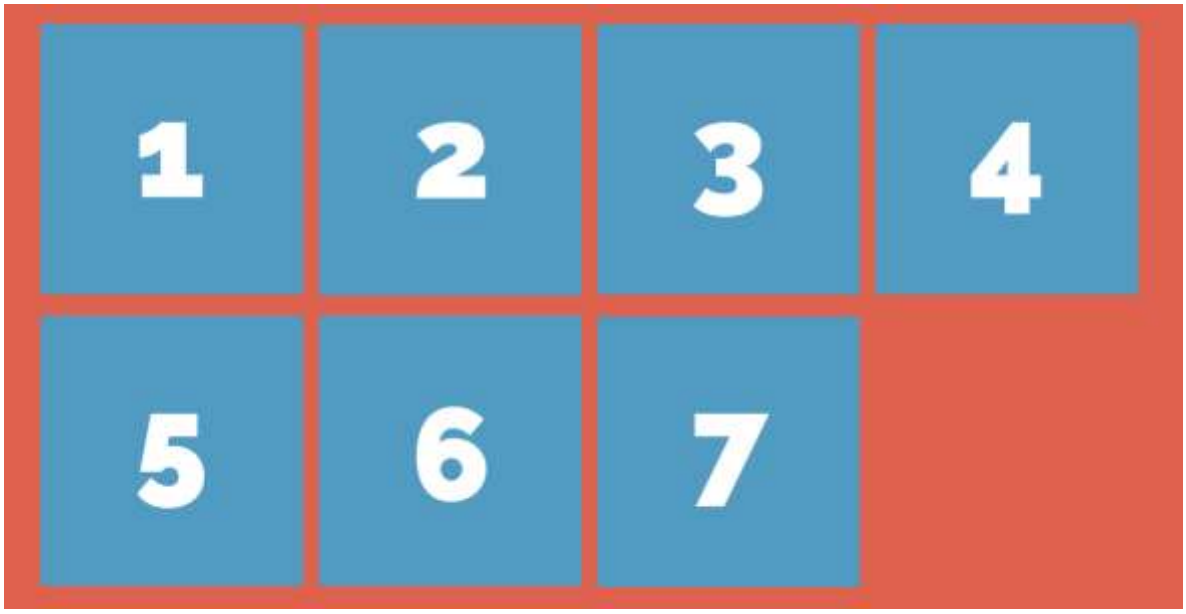
En manejar Flexbox el concepte inicial és fixar els **elements - flex** en una sola línia, però amb la propietat flex-wrap controlem si el **contenidor flex** mou els seus **elements - flex** en línies individuals o múltiples, és una de les coses més sorprenents de FlexBox, ja que amb una simple línia de codi ens estalviem molts problemes.

```
.pare { flex-wrap: nowrap; }
```

Els **elements - flex** es mostren en una fila i es poden encongir depenent del seu **contenidor - flex.**

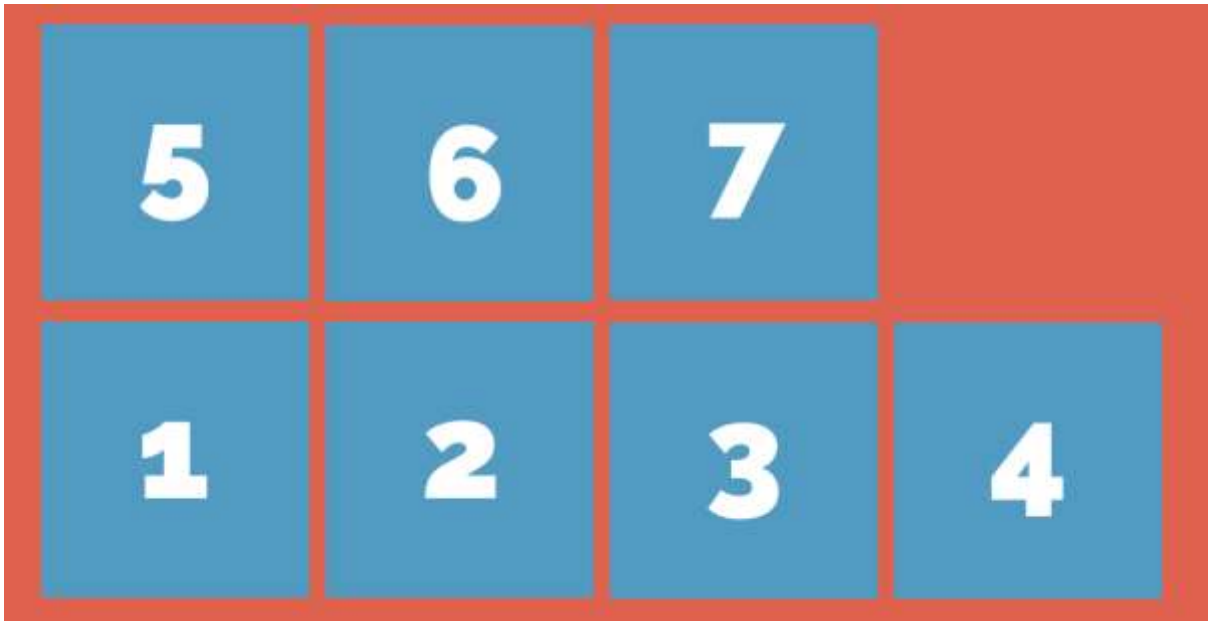
```
.pare { flex-wrap: wrap; }
```

Els **elements - flex** es mostren en diverses files (si cal), d'esquerra a dreta i de dalt a baix.




```
.pare { flex-wrap: wrap-reverse; }
```

Els **elements - flex** es mostren en diverses files (si cal), d'esquerra a dreta i de baix a dalt.



FLEX-FLOW

Valor per defecte: row nowrap

Aquesta propietat és una forma ràpida per establir les propietats flex-direction i flex-wrap .

```
.pare { flex-flow: <flex-direction> || <flex-wrap>; }
```

exemple:

```
.pare { flex-flow: column-reverse wrap; }
```

JUSTIFY-CONTENT

Valor per defecte: flex-start

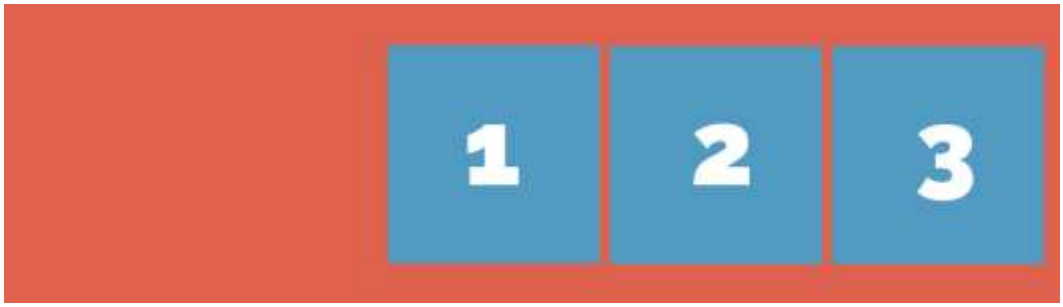
La propietat justify-content alinea els **elements - flex** al llarg de **l'eix principal** de la línia actual **(contenidor - flex).**

```
.pare { justify-content: flex-start; }
```

Els **elements - flex** estan alineats amb el costat esquerre del **contenidor - flex.**


```
.pare { justify-content: flex-end; }
```

Els **elements - flex** estan alineats amb el costat dret del **contenidor - flex.**




```
.pare { justify-content: center; }
```

Els **elements - flex** estan alineats al centre del **contenidor - flex.**


```
.pare { justify-content: space-between; }
```

Els **elements - flex** tenen la mateixa distància entre ells, però el primer i l'últim **element – flex** estan alineats amb les vores del **contenidor - flex.**


```
.pare { justify-content: space-around; }
```

Els **elements - flex** tenen la mateixa distància entre ells, fins i tot el primer i l'últim **element - flex.**

ALIGN-ITEMS

Valor per defecte: stretch

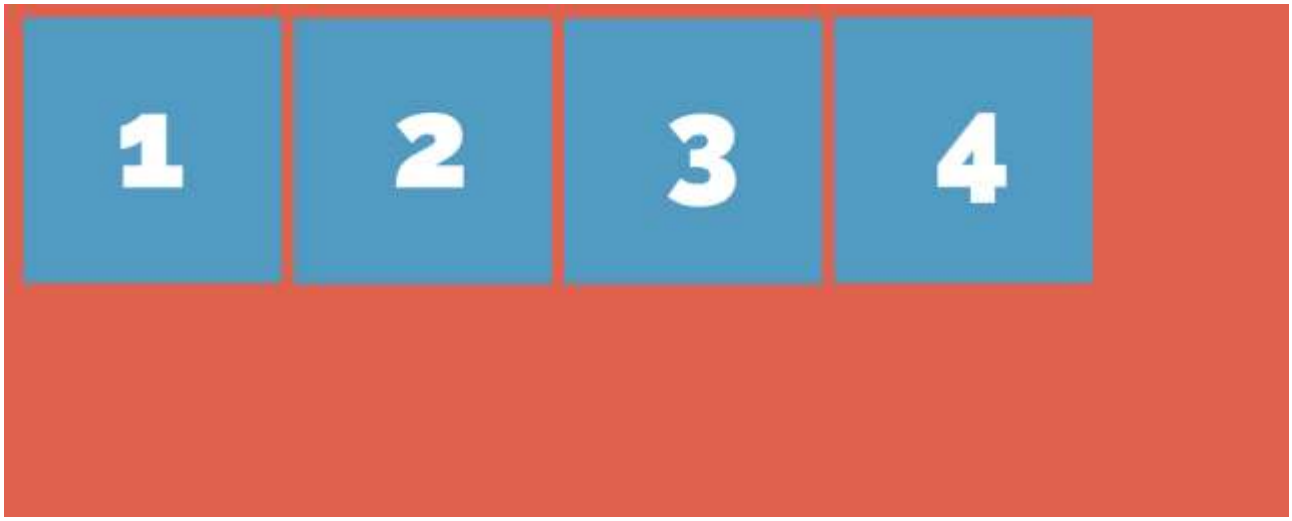
La propietat `align-items` és molt similar a la propietat `justify-content`, però aquest és en la direcció de **l'eix secundari.**

```
.pare { align-items: stretch; }
```

Els **elements - flex** ocupen tota l'alçada (o amplada) **d'inici secundari** a **fi secundari.**

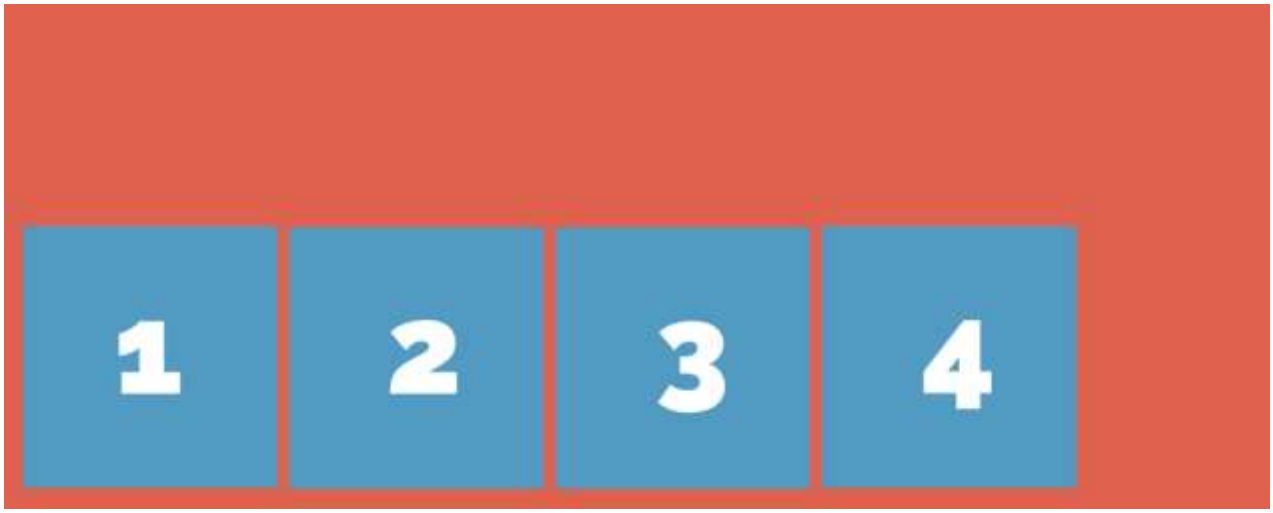

```
.pare { align-items: flex-start; }
```

Els **elements - flex** s'apilen a **l'inici secundari** del **contingut - flex.**



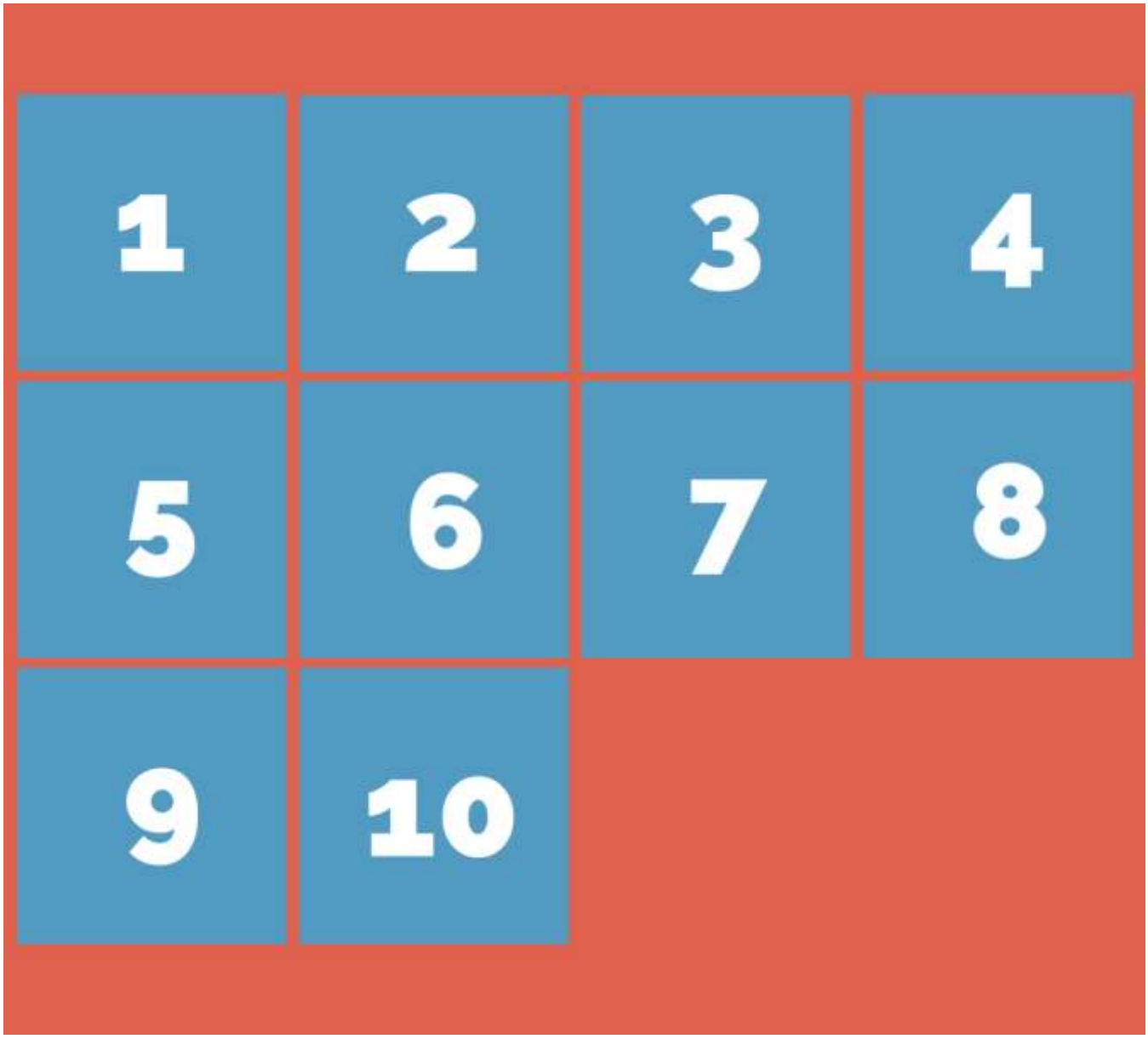

```
.pare { align-items: flex-end; }
```

Els **elements - flex** s'apilen a la **fi secundari** del **contingut - flex.**



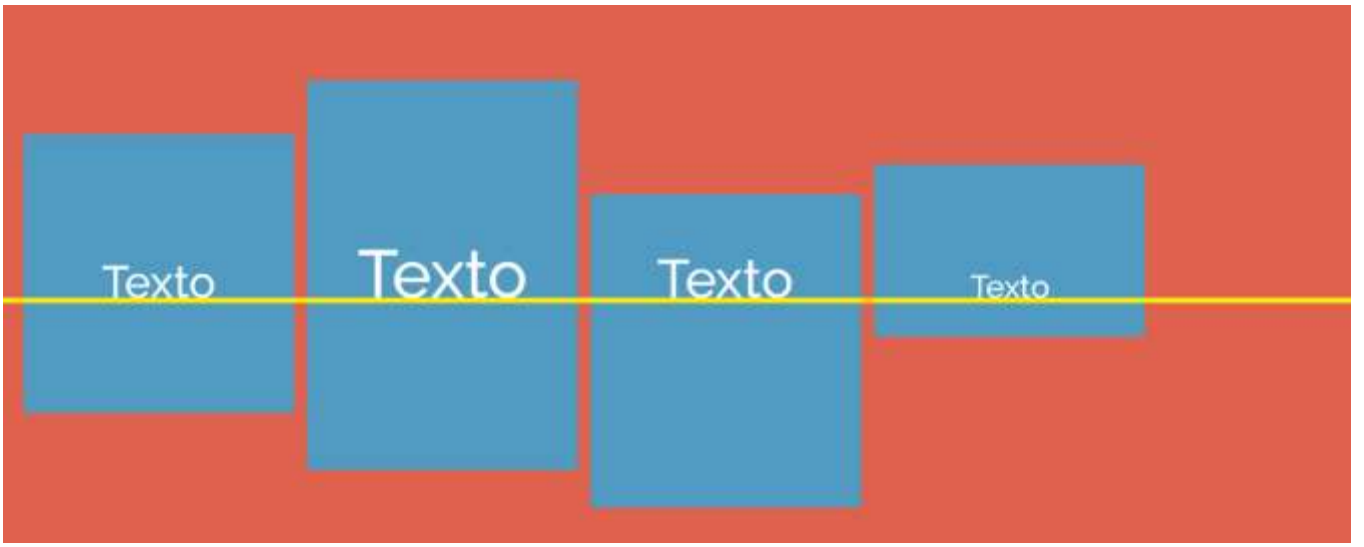

```
.pare { align-items: center; }
```

Els **elements - flex** s'apilen al centre de **l'eix secundari.**




```
.pare { align-items: baseline; }
```

Els **elements - flex** s'alineen de manera que les seves línies base queden iguals.



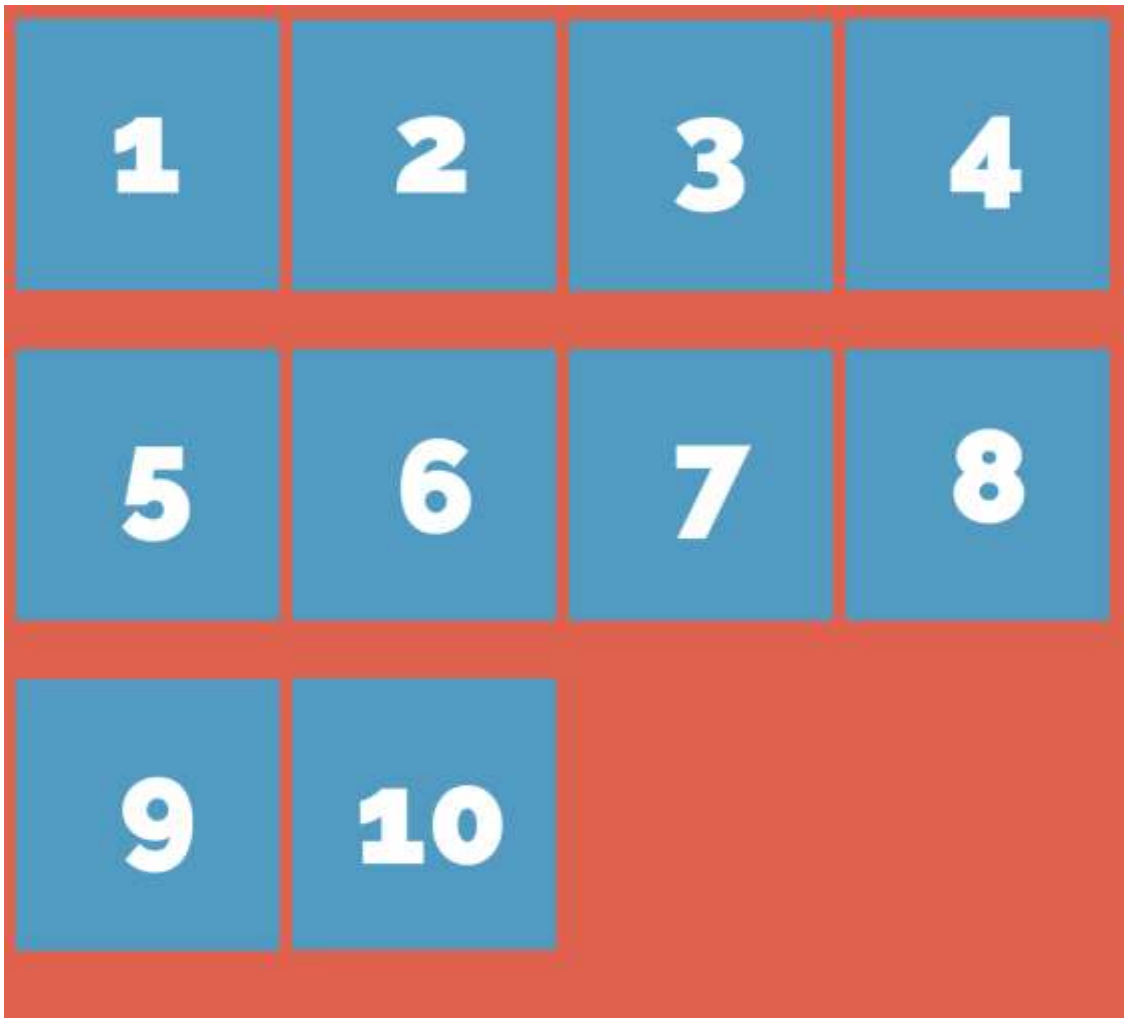
ALIGN-CONTENT

Valor per defecte: stretch

La propietat align-content alinea les línies d'un **contenidor - flex** quan els elements no utilitzen tot l'espai disponible a **l'eix secundari.**

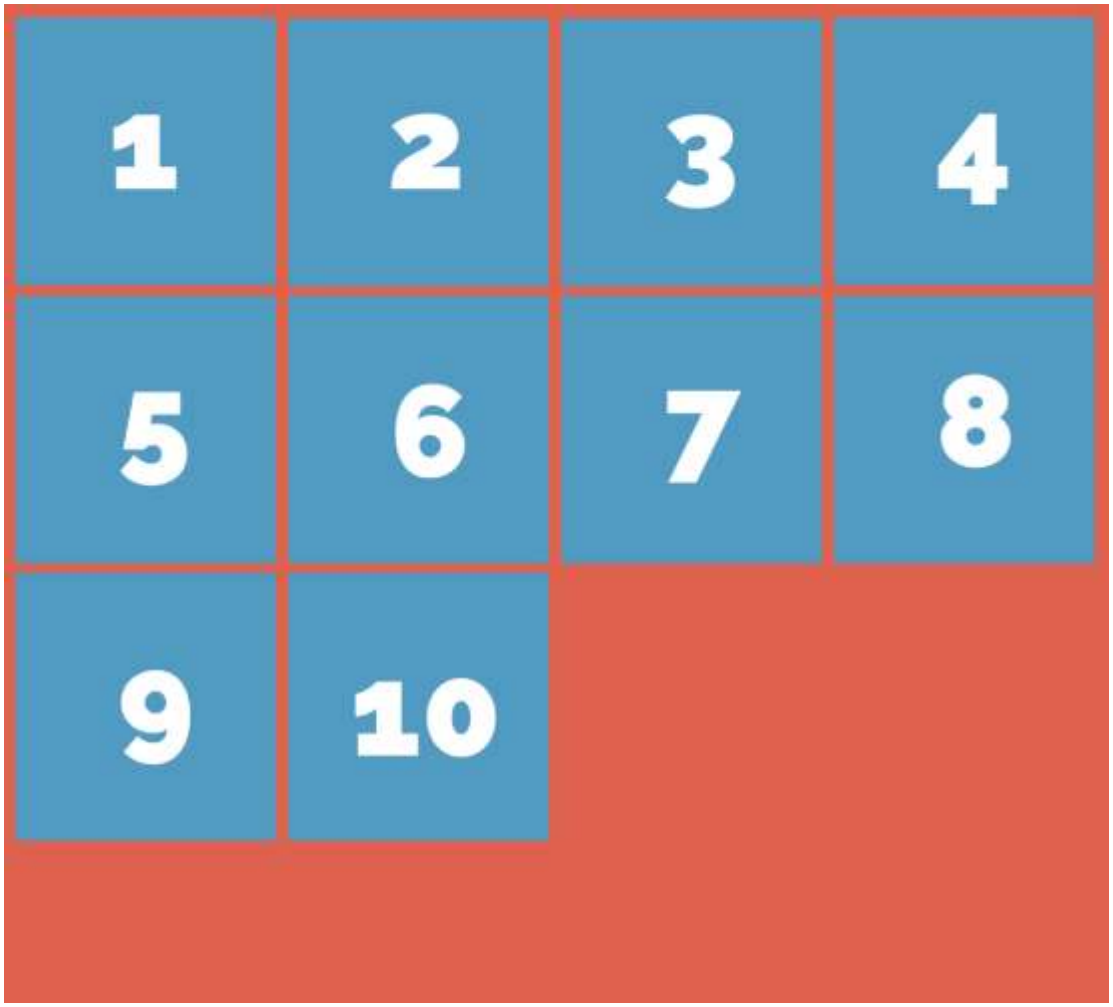
```
.pare { align-content: stretch; }
```

Els **elements - flex** es mostren amb espai distribuït després de cada fila **d'elements - flex.**



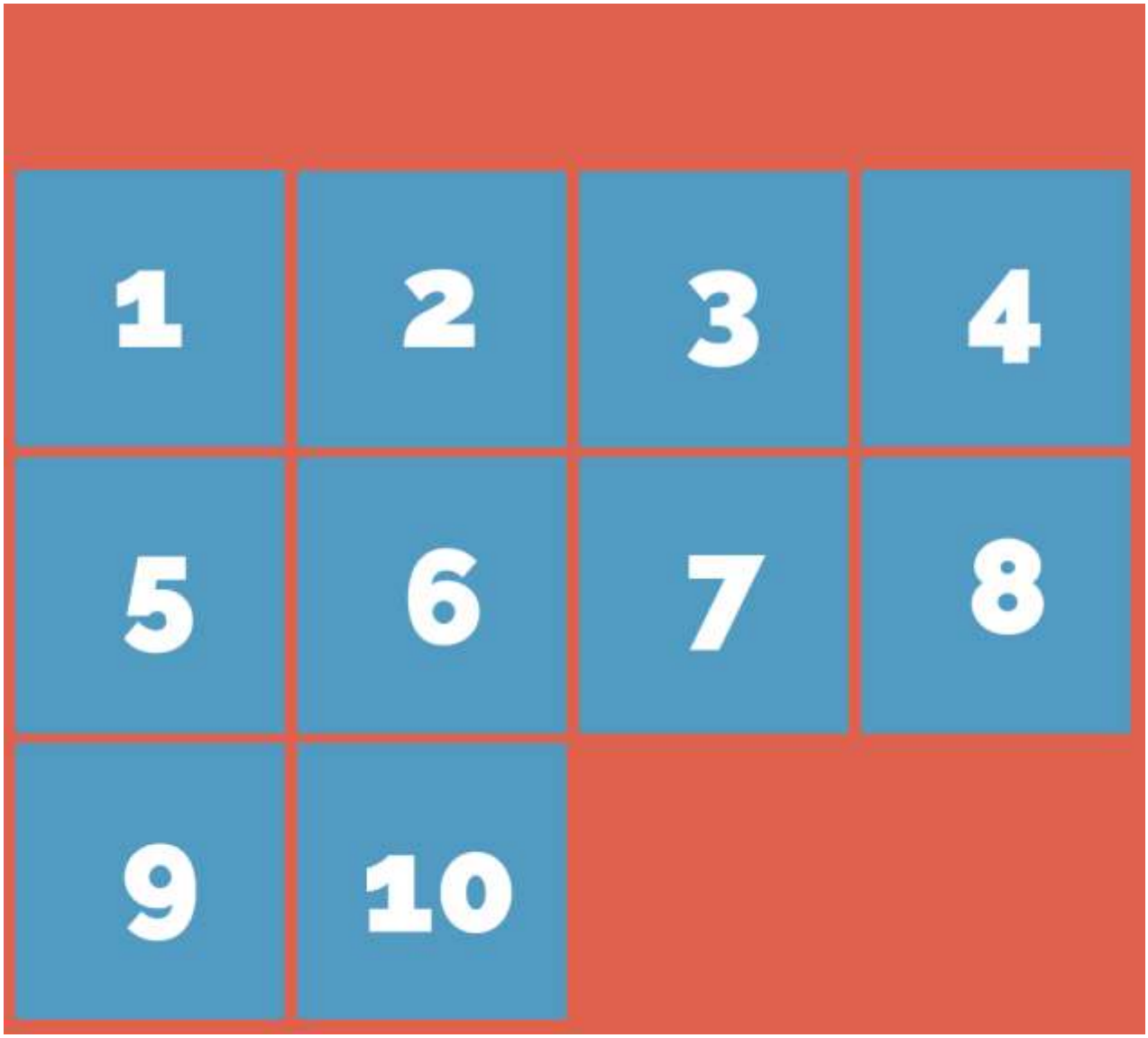

```
.pare { align-content: flex-start; }
```

Els **elements - flex** s'apilen cap a **l'inici secundari** del **contenidor - flex.**



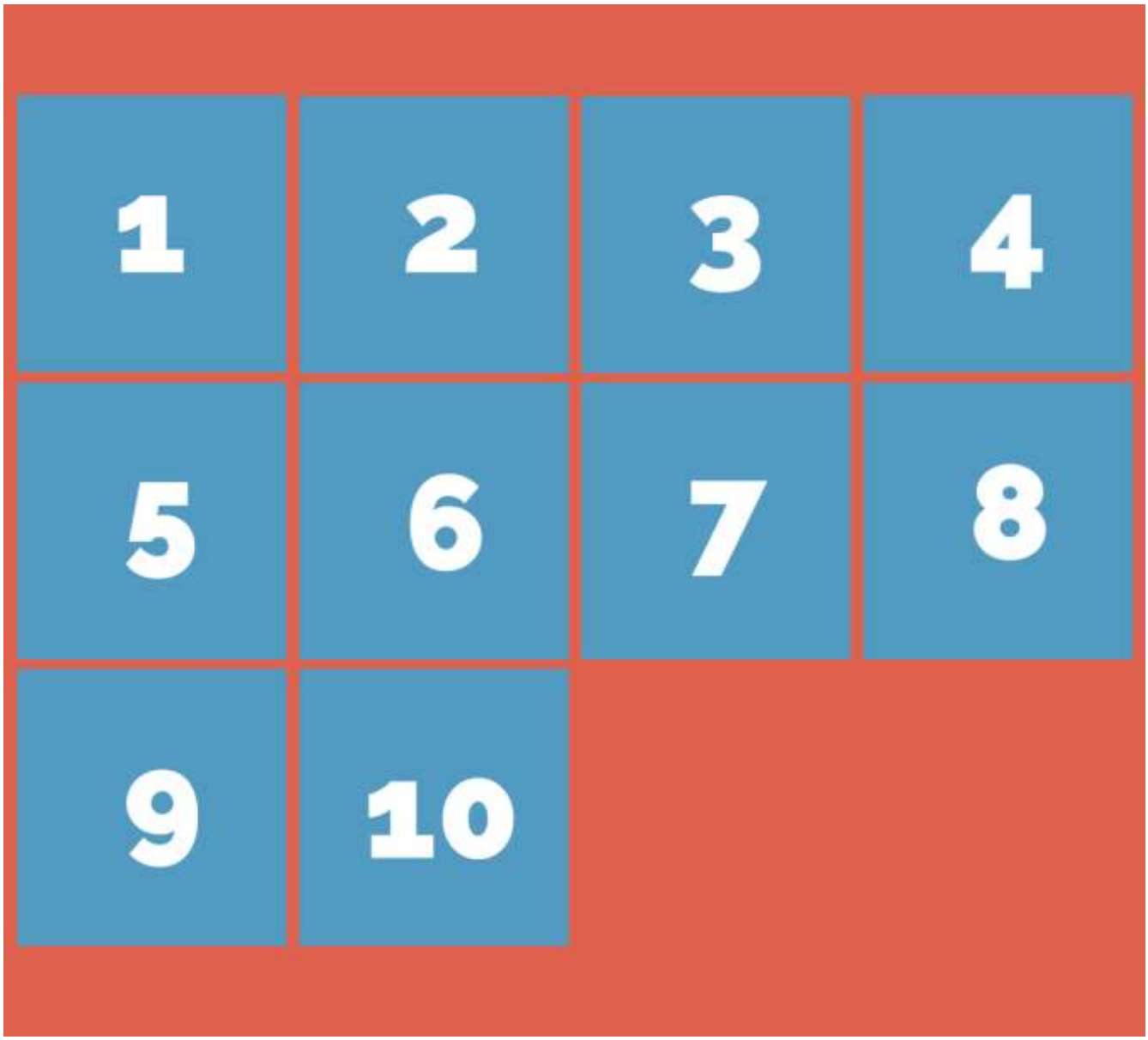

```
.pare { align-content: flex-end; }
```

Els **elements - flex** s'apilen cap al **final secundari** del **contenidor - flex.**



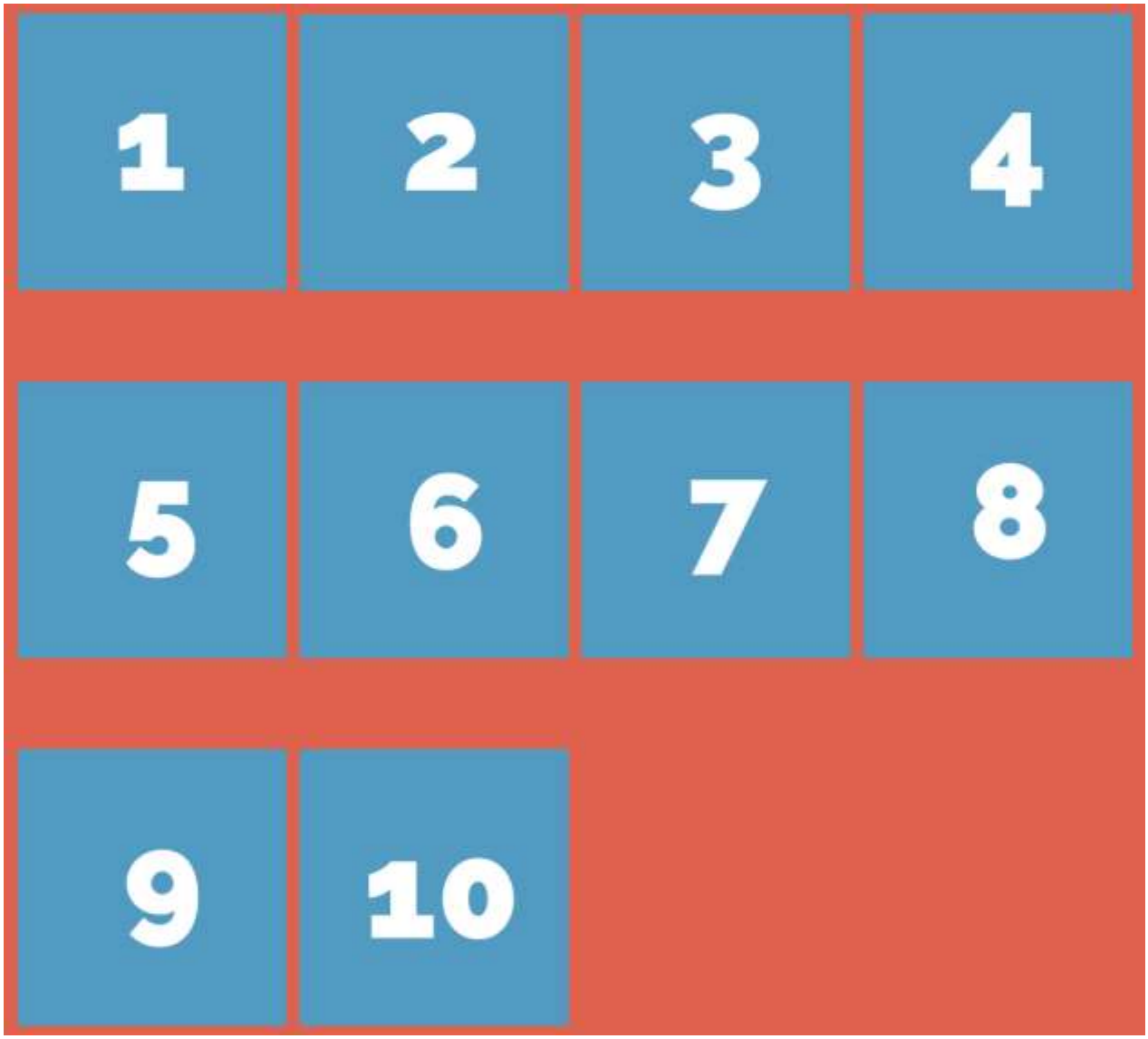

```
.pare { align-content: center; }
```

Les files s'apilen al centre de **l'eix secundari.**



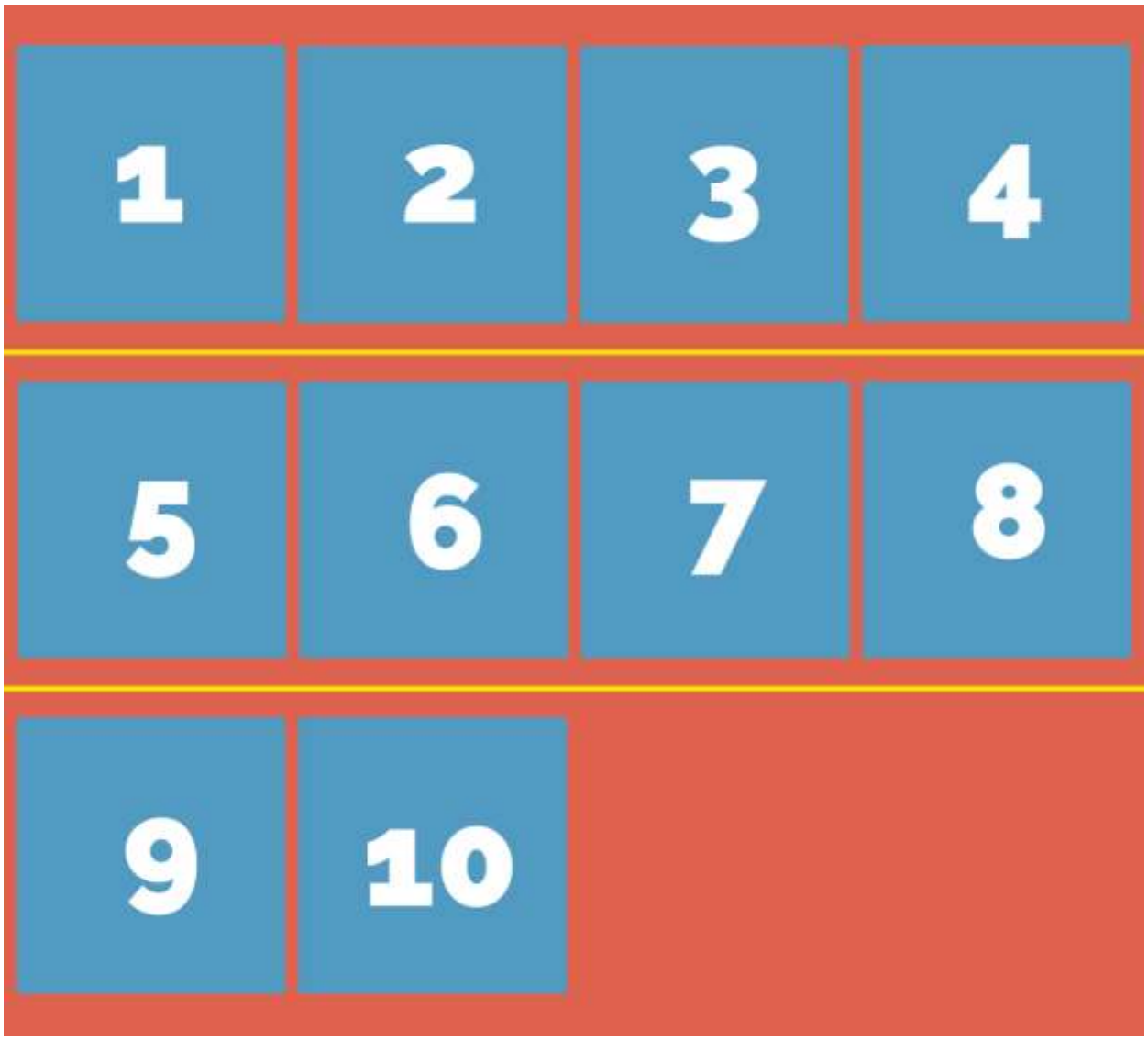

```
.pare { align-content: space-between; }
```

Les files dels **elements - flex** tenen la mateixa separació entre ells, però la primera i última fila estan alineades amb les vores del **contenidor - flex.**




```
.pare { align-content: space-around; }
```

Els **elements - flex** tenen la mateixa separació en cada fila **d'elements - flex.**



Nota: Aquesta propietat només funciona quan el **contenidor - flex** té diverses línies **d'elements - flex.** Si només hi ha una sola línia aquesta propietat no té efecte.

Gràcies a aquestes propietats ja podem crear coses increïbles i fantàstiques, però compte!, això no és tot, encara falten més propietats, només que aquestes no són pel contenidor (Pare), sino, que aquestes s'apliquen per als elements (fills).

Propietats dels elements - flex (Fills)

Igual que el **contenidor - flex,** els **elements - flex** tenen propietats interessants, amb les quals podrem fer coses encara més increïbles.

ORDER

Valor per defecte: 0

Amb aquesta propietat controlem l'ordre dels **elements - flex** que són dins el **contenidor - flex.**

```
.fill { order: <integer>; }
```

Els **elements - flex** poden ser reordenats amb aquesta simple propietat sense la necessitat de reestructurar el codi HTML.

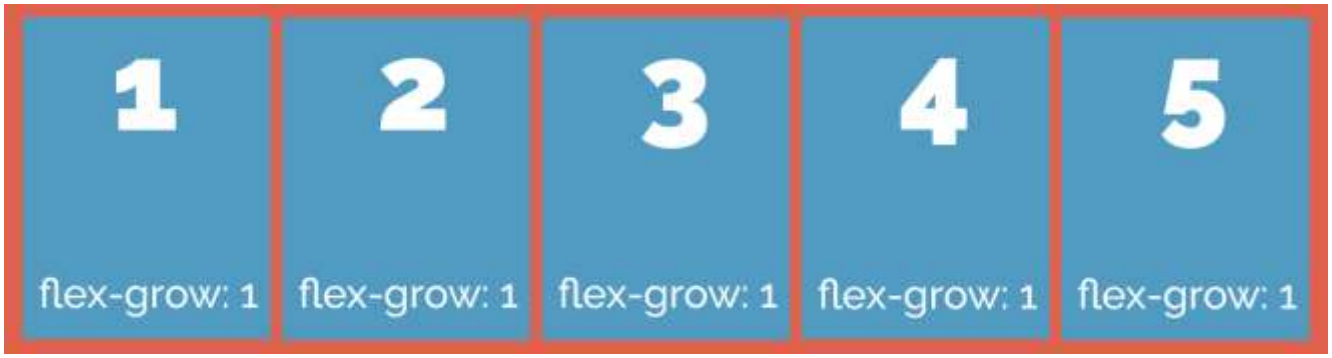
FLEX-GROW

Valor per defecte: 0

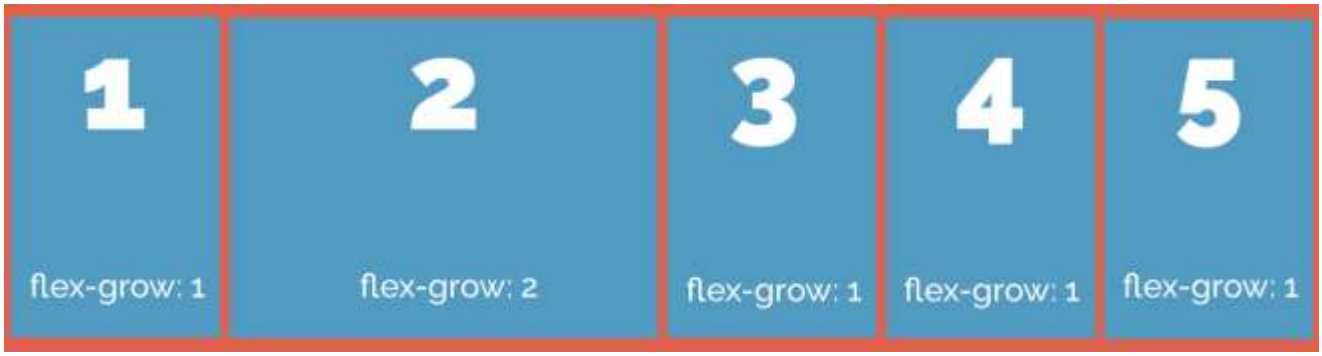
Aquesta propietat especifica el factor que determina fins a quin punt un **element – flex** creixerà en relació amb la resta dels **elements - flex.**

```
.fill { flex-grow: <number>; }
```

Si els **elements - flex** tenen el mateix valor per `flex-grow` , tots tindran la mateixa mida, segons el **contenidor - flex.**



En definir que en el segon **element - flex** la propietat `flex-grow` tindrà un valor de **dos,** aquest va a prendre un ample que equival a la suma de dos elements.



Nota: Els nombres negatius igual s'apliquen per a aquesta propietat.

FLEX-SHRINK

Valor per defecte: 1

Aquesta propietat especifica com **l'element - flex** es reduirà respecte a la resta dels elements flexibles dins del mateix contenidor.

```
.fill{ flex-shrink: <number>; }
```

Per defecte, tots els **elements - flex** es poden reduir, però si li posem el valor de 0 no s'encongiran, si no, que mantindran la mida original.

Nota: Els nombres negatius igual apliquen per a aquesta propietat.

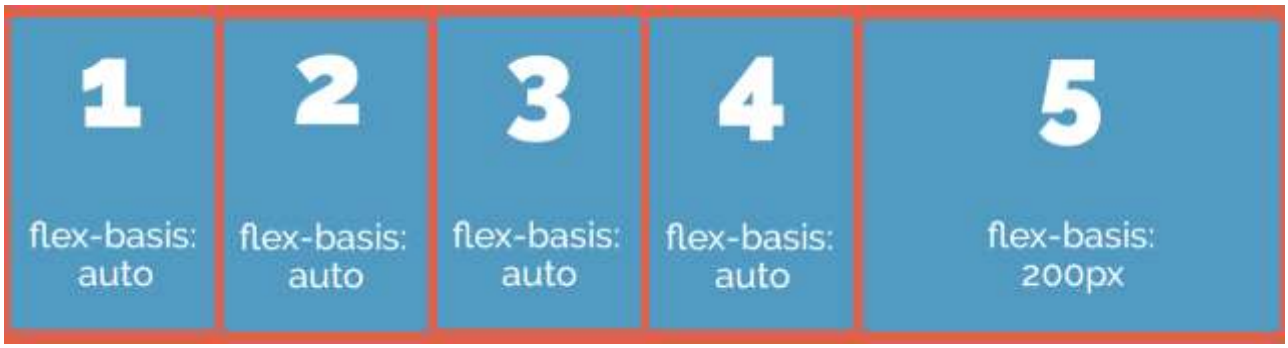
FLEX-BASIS

Valor per defecte: auto

Aquesta propietat té els mateixos valors que les propietats width i height i especifica la mida principal de **l'element - flex,** distribuint-se dacord amb els factors **flex.**

```
.fill { flex-basis: auto | <width>; }
```

Aquí especifiquem que el cinquè **element - flex** dicta la mida inicial de l'element.



FLEX

Valor per defecte: 0 1 auto

Amb aquesta propietat només és l'abreviatura de flex-grow , flex-shrink i flex-basis . Entre d'altres valors que també es poden configurar per auto (1 1 auto) i none (0 0 auto).

```
.fill { flex: none | auto | [ <flex-grow> <flex-shrink>? || <flex-basis> ]; }
```

ALIGN-SELF

Valor per defecte: auto

Aquesta propietat permet l'alineació per defecte (o l'especifica per align-items) per a ser anul·lats per **elements - flex** individuals.

```
.fill { align-self: auto | flex-start | flex-end | center | baseline | stretch; }
```

El tercer i cinquè **element - flex** han anul·lat l'alineació a través de la propietat align-self .

Nota: float , clear i vertical-align no tenen efecte en un **element - flex.**

FLEXBOX Y DISTRIBUCIÓ DE LA PÀGINA HTML5

Anem a provar una mica millor jugant amb flexibilitat d'elements flexibles. Què passa amb un disseny de 3 columnes amb un encapçalament i un peu de pàgina d'ample complet. I independent del codi font.

```
.wrapper {
  display: flex;
  flex-flow: row wrap;
}
/* We tell all items to be 100% width */
.header, .main, .nav, .aside, .footer {
  flex: 1 100%;
}
/* We rely on source order for mobile-first approach
 * in this case:
 * 1. header
 * 2. nav
 * 3. main
 * 4. aside
 * 5. footer
 */

/* Medium screens */
@media all and (min-width: 600px) {
  /* We tell both sidebars to share a row */
  .aside { flex: 1 auto; }
}
/* Large screens */
@media all and (min-width: 800px) {
  /* We invert order of first sidebar and main
   * And tell the main element to take twice as much width as the other two sidebars
   */
  .main { flex: 2 0px; }

  .aside-1 { order: 1; }
  .main    { order: 2; }
  .aside-2 { order: 3; }
  .footer  { order: 4; }
```

El resultat

Bibliografia

Bibliografia:

https://filisantillan.com/el-gran-poder-de-css3-flexbox/

https://css-tricks.com/snippets/css/a-guide-to-flexbox/