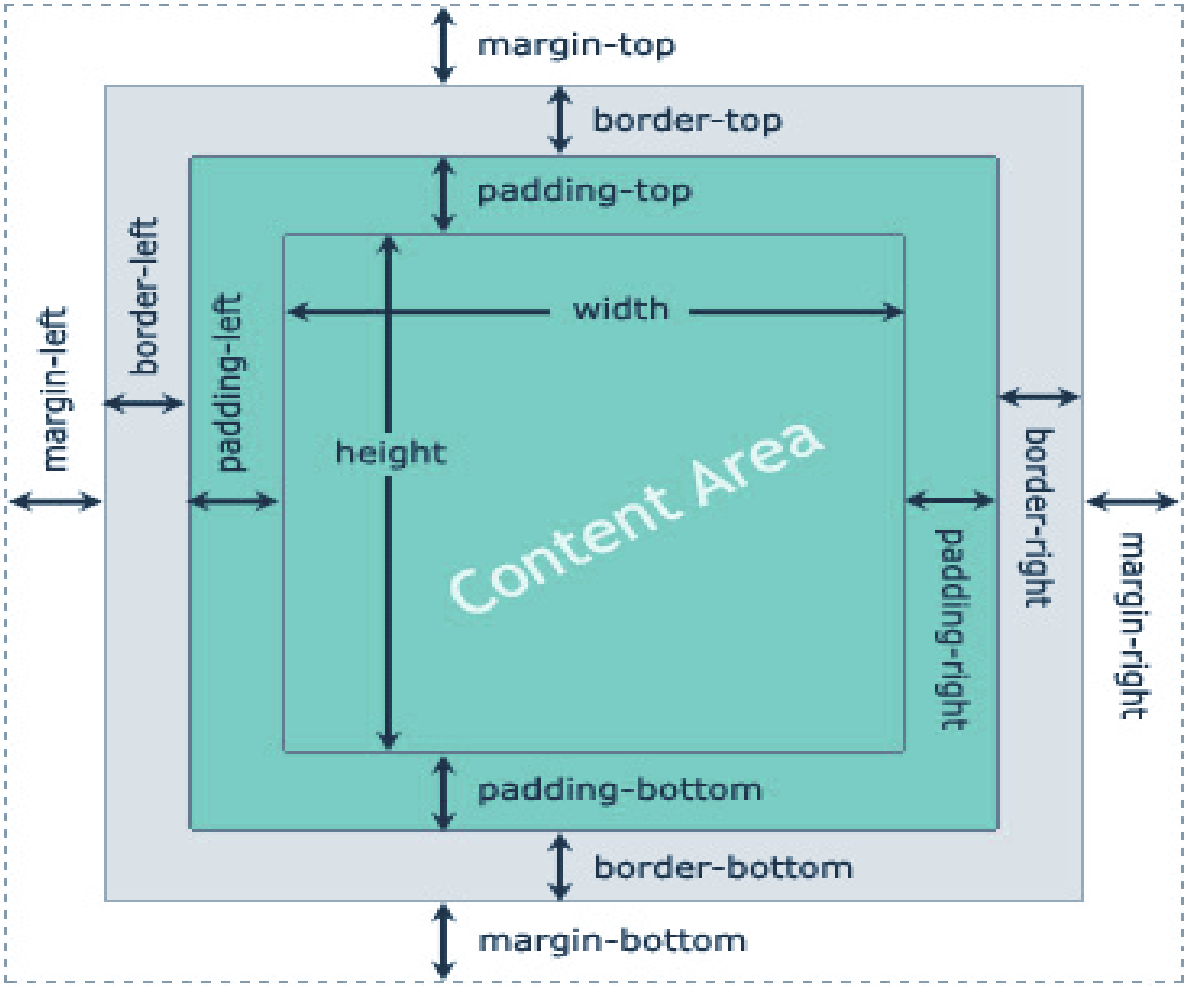
Flexbox

FlexBox

- Flexbox: modalità di layout introdotta con CSS3
- L'uso dei Flexbox garantisce un comportamento più predicibile e stabile degli elementi della pagina, sopratutto nel caso di layout liquido e responsive design, quando la pagina deve adattarsi a display di diversi device e diverse dimensioni
- Rappresenta una innovazione ed un notevole miglioramento nella gestione dei layout rispetto al box model tradizionale

Box Model


margin-top
border-top
padding-top
margin-left
border-left
padding-left
width
height
Content Area
padding-right
border-right
margin-right
padding-bottom
border-bottom
margin-bottom

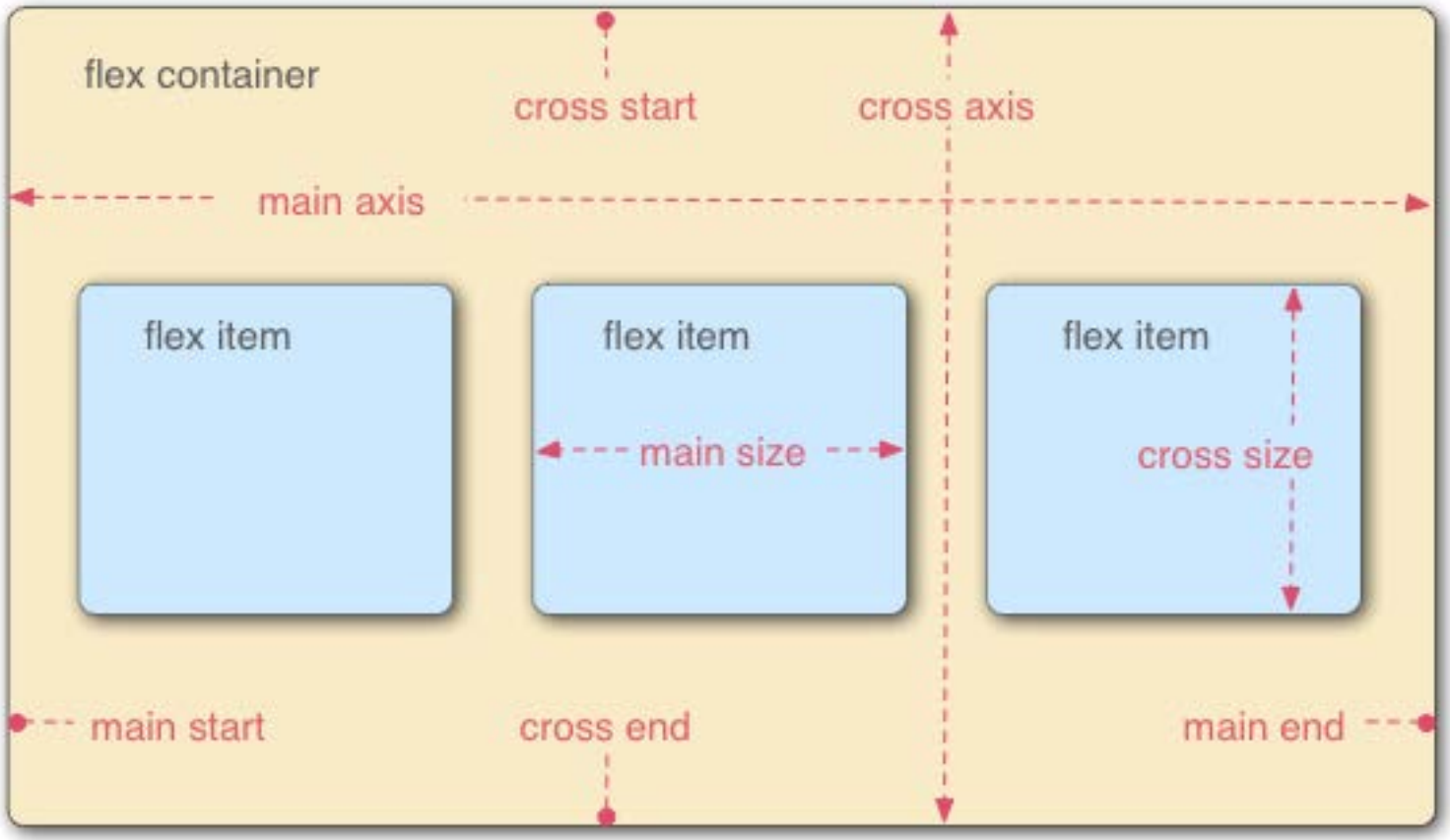
Concetti principali dei Flexbox

- Gli elementi alla base del modello flexbox sono:
  - ***Flex container***: viene dichiarato impostando la proprietà "***display***" di un elemento, che potrebbe avere valore "***flex***" (reso come un elemento di blocco) o "***inline-flex***" (reso come un elemento inline)
  - ***Flex item***: all'interno di un ***flex container*** possono essere posizionati uno o più ***flex item***

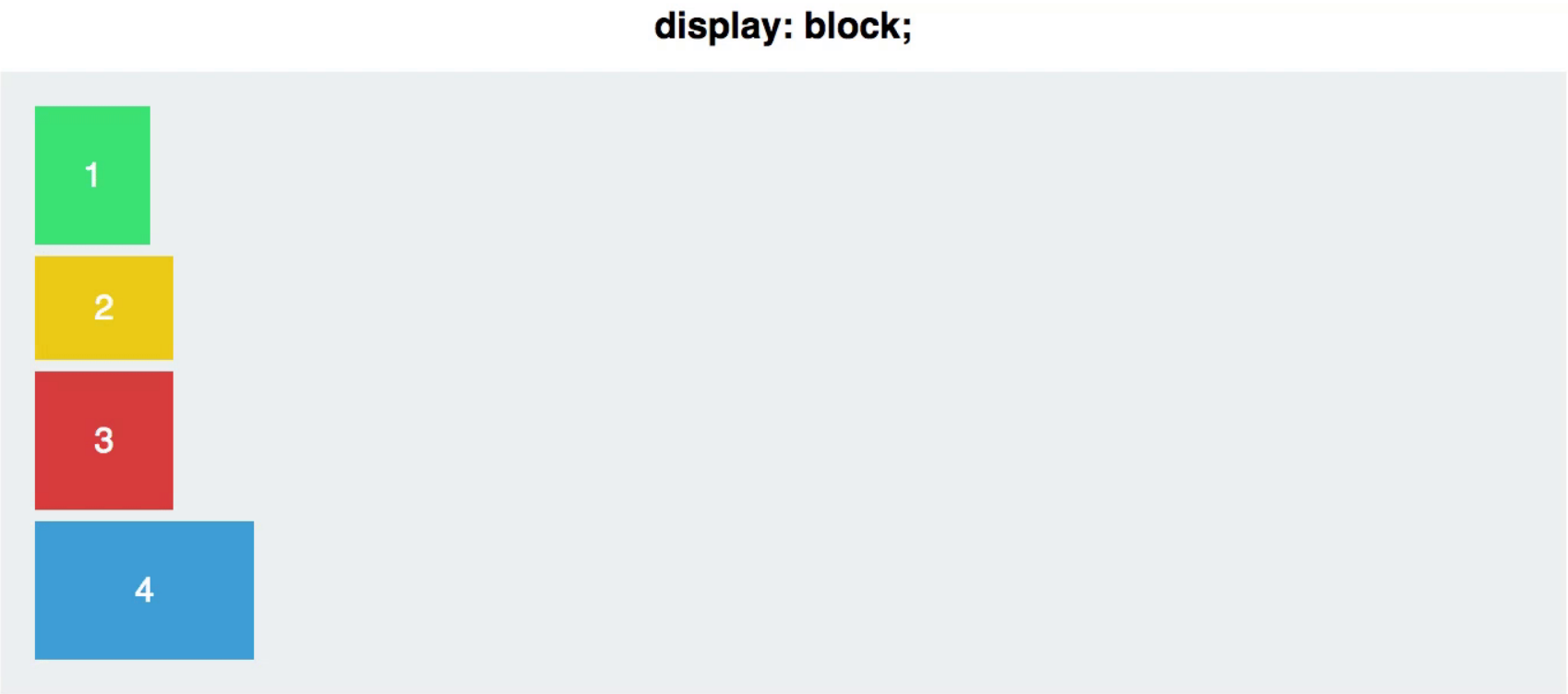
Concetti principali

- Gli elementi all'estero di un ***flex-container*** e gli elementi all'interno di un ***flex-item*** sono resi con il box model tradizionale
- Il modello flexbox definisce come i ***flex item*** si dispongono all'interno di un ***flex container***
- I ***flex item*** sono posizionati all'interno di un ***flex container*** lungo una ***flex line***
- La situazione di default consiste nell'avere una sola ***flex line*** per ogni ***flex container***

Flexbox Model



Display: block; vs Display: flex;



Un esempio

- In questo esempio vediamo 3 ***flex item***
- Il loro posizionamento è quello di default: lungo la ***flex line*** orizzontale, da sinistra a destra

[Gli esempi di questa lezione sono raggruppati nello zip “esempi_flexbox” presente sulla piattaforma]

- **`flex_item_01.html`**

flex_item_01.html

```
<!DOCTYPE html>
<html>
<head>
</head>
<body>

<div class="flex-container">
  <div class="flex-item">
        flex item 1
  </div>
  <div class="flex-item">
        flex item 2
  </div>
  <div class="flex-item">
        flex item 3
  </div>
</div>

</body>
</html>
```

```
body {
    font-family: arial;
    color: white;
}

.flex-container {
    display: flex;
    width: 50%;
    height: 250px;
    background-color: grey;
}

.flex-item {
    background-color: darkred;
    width: 30%;
    height: 100px;
    margin: 10px;
    padding: 10px;
}
```

direction

- E’ possibile modificare la direzione della ***flex line***
- Se impostiamo la proprietà ***direction*** a ***rtl*** (right-to-left), allora il testo sarà mostrato da destra a sinistra e la ***flex line*** cambierà direzione, modificando il layout della pagina
- **`direction_02.html`**

direction_02.html

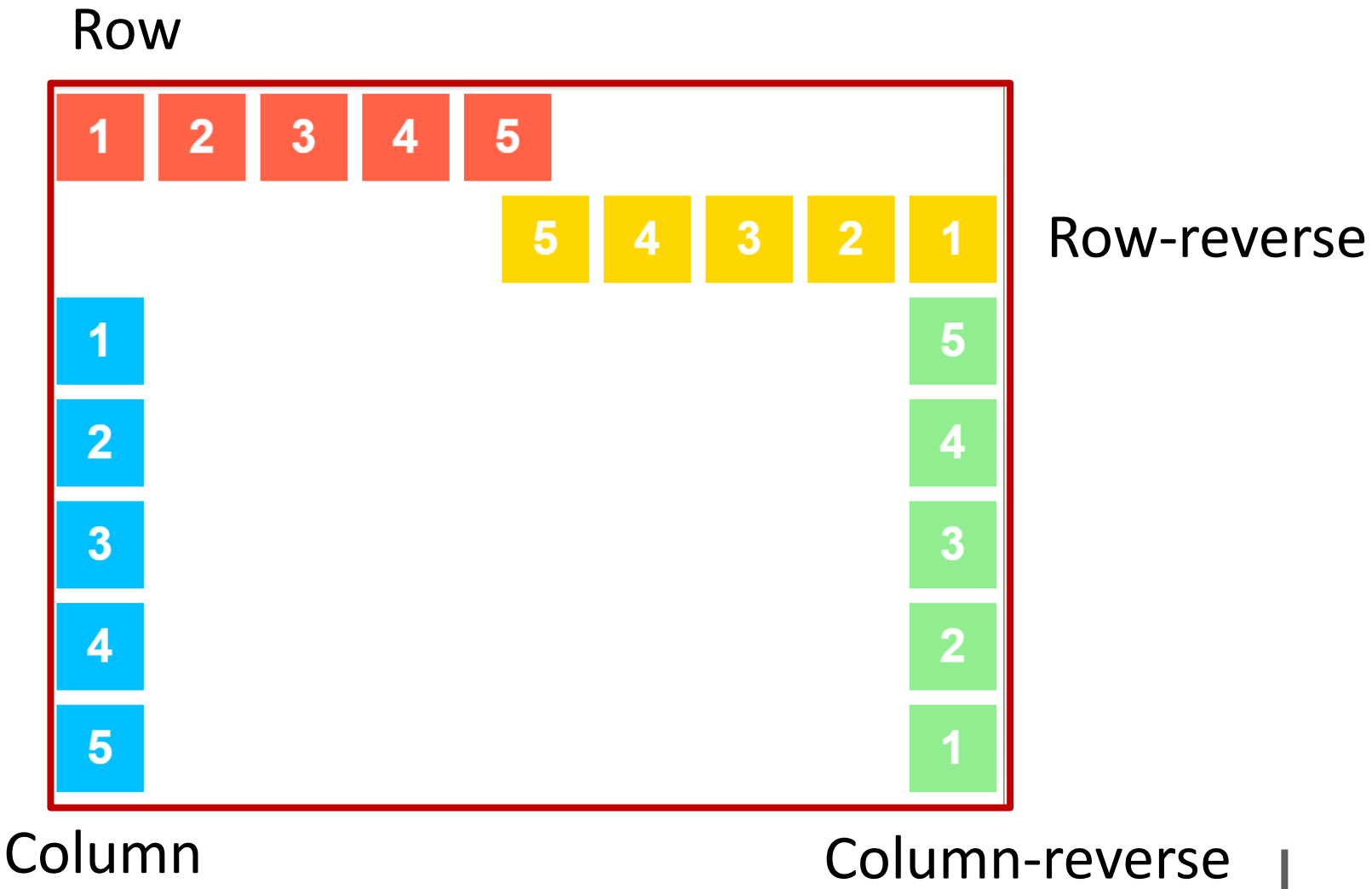
```
<!DOCTYPE html>
<html>
<head>
</head>
<body>

<div class="flex-container">
  <div class="flex-item">
        flex item 1
  </div>
  <div class="flex-item">
        flex item 2
  </div>
  <div class="flex-item">
        flex item 3
  </div>
</div>

</body>
</html>
```

```
body {
    direction: rtl;
    font-family: arial;
    color: white;
}

.flex-container {
    display: flex;
    width: 50%;
    height: 250px;
    background-color: grey;
}

.flex-item {
    background-color: darkred;
    width: 30%;
    height: 200px;
    margin: 10px;
    padding: 10px;
}
```

Flex Direction

- La proprietà ***flex-direction*** permette di specificare la direzione dei ***flex item*** all'interno del ***flex container***
- Il valore di default per ***flex-direction*** è ***row*** (left-to-right, top-to-bottom)
- Altri valori:
  - **row-reverse**
  - **column**
  - **column-reverse**

Flex Direction



Flex Direction: row-reverse

- **row-reverse** – se la direzione della scrittura del testo (direction) è da sinistra a destra, allora il ***flex item*** sarà disposto nella direzione opposta, ovvero da destra a sinistra
- Il prossimo esempio mostra il risultato dell'uso del valore **row-reverse**:
  `flex_direction_03.html`

flex_direction_03.html

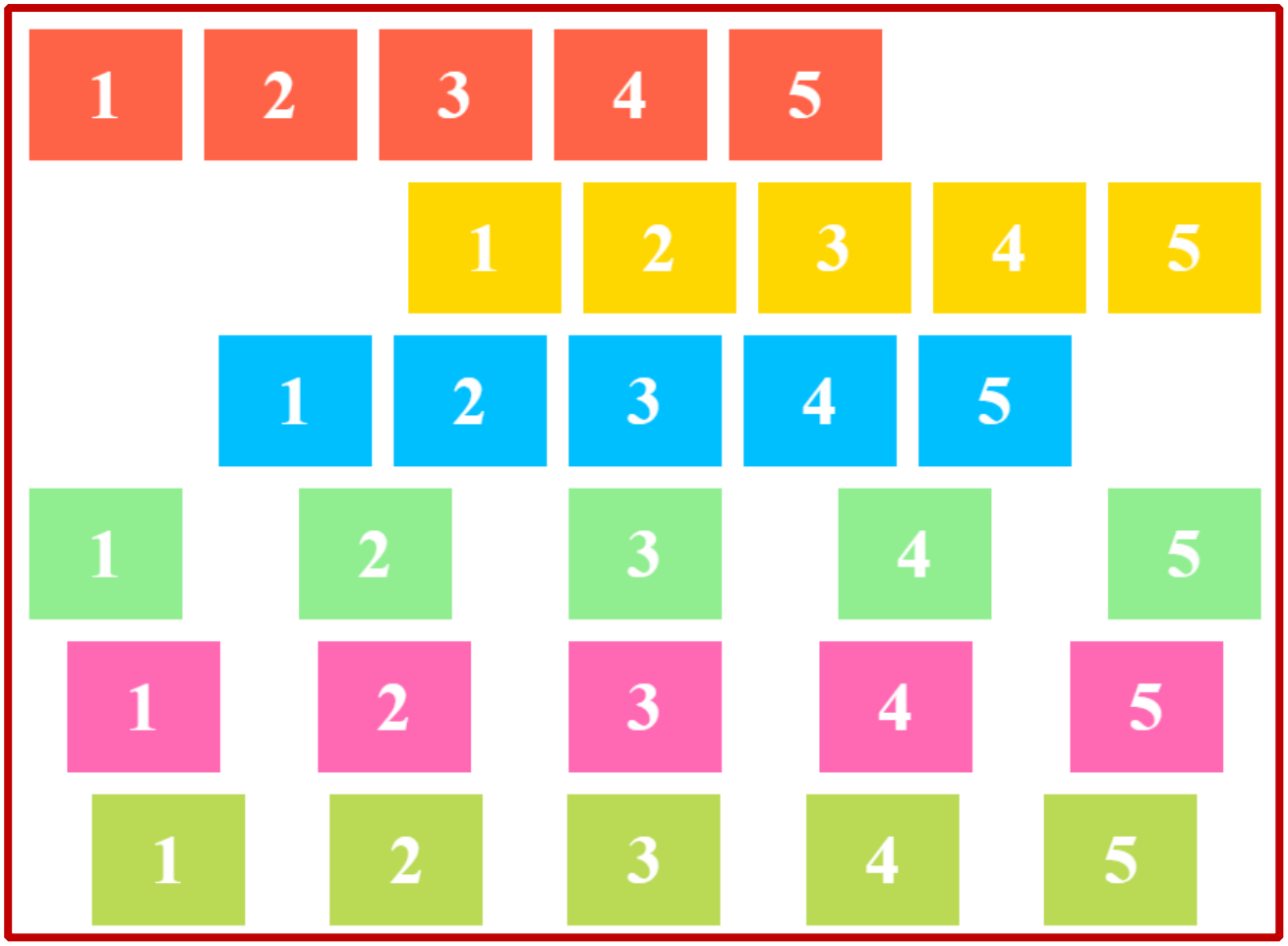
```
<!DOCTYPE html>
<html>
<head>
</head>
<body>

<div class="flex-container">
  <div class="flex-item">
        flex item 1
  </div>
  <div class="flex-item">
        flex item 2
  </div>
  <div class="flex-item">
        flex item 3
  </div>
</div>

</body>
</html>
```

```
body {
    font-family: arial;
    color: white;
}

.flex-container {
    display: flex;
    flex-direction: row-reverse;
    width: 50%;
    height: 250px;
    background-color: grey;
}

.flex-item {
    background-color: darkred;
    width: 20%;
    height: 100px;
    margin: 10px;
    padding: 10px;
}
```

Justify-content

- La proprietà ***justify-content*** allinea orizzontalmente gli elementi di un ***flexible container*** quando questi elmenti non utilizzano tutto lo spazio a disposizione lungo l'asse delle ascisse
- Il valore di default è ***flex-start***: gli elementi sono posizionati all'inizio del container
- Altri possibili valori sono i seguenti:
  - ***flex-end***
  - ***center***
  - ***space-between***
  - ***space-around***
  - ***space-evenly*** (simile a ***space-around*** ma lo spazio tra i flex-item e il bordo del container è distribuito in modo equilibrato)

Justify-content



Flex-start

Flex-end

Center

Space-between

Space-around

Space-evenly

Justify-content: flex-end

- **flex-end**: gli item sono posizionati alla fine del container
- Il prossimo esempio mostra il risultato dell'uso del valore **flex-end**:
  `justify_content_04.html`

justify_content_04.html

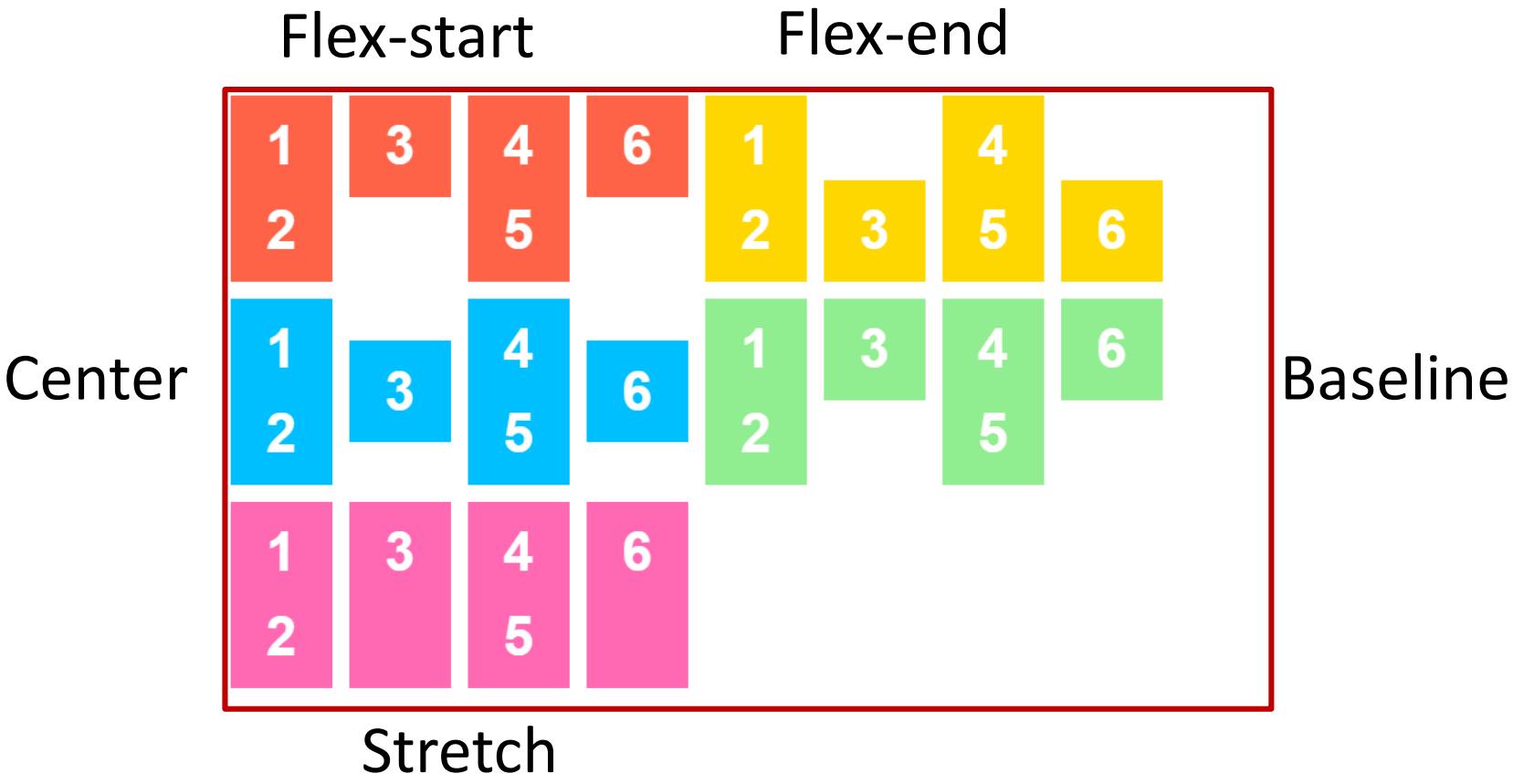
```
<!DOCTYPE html>
<html>
<head>
</head>
<body>

<div class="flex-container">
  <div class="flex-item">
        flex item 1
  </div>
  <div class="flex-item">
        flex item 2
  </div>
  <div class="flex-item">
        flex item 3
  </div>
</div>

</body>
</html>
```

```
body {
    font-family: arial;
    color: white;
}

.flex-container {
    display: flex;
    justify-content: flex-end;
    width: 50%;
    height: 250px;
    background-color: grey;
}

.flex-item {
    background-color: darkred;
    width: 20%;
    height: 100px;
    margin: 10px;
    padding: 10px;
}
```

Align-items

- La proprietà ***align-items*** allinea verticalmente gli elementi del ***flex container*** quando gli item non usano tutto lo spazio disponibile lungo l'asse delle ordinate
- Il valore di default è **stretch**: gli item vengono "stretchati" per riempire opportunamente il ***container***
- Gli altri valori ammessi sono i seguenti:
  - **flex-start**
  - **flex-end**
  - **center**
  - **baseline**

Align-items



Align-items: center

- **center**: gli item sono posizionati al centro del container (verticalmente)
- Il prossimo esempio mostra il risultato dell'uso del valore **center**:
  `align_items_05.html`

align_items_05.html

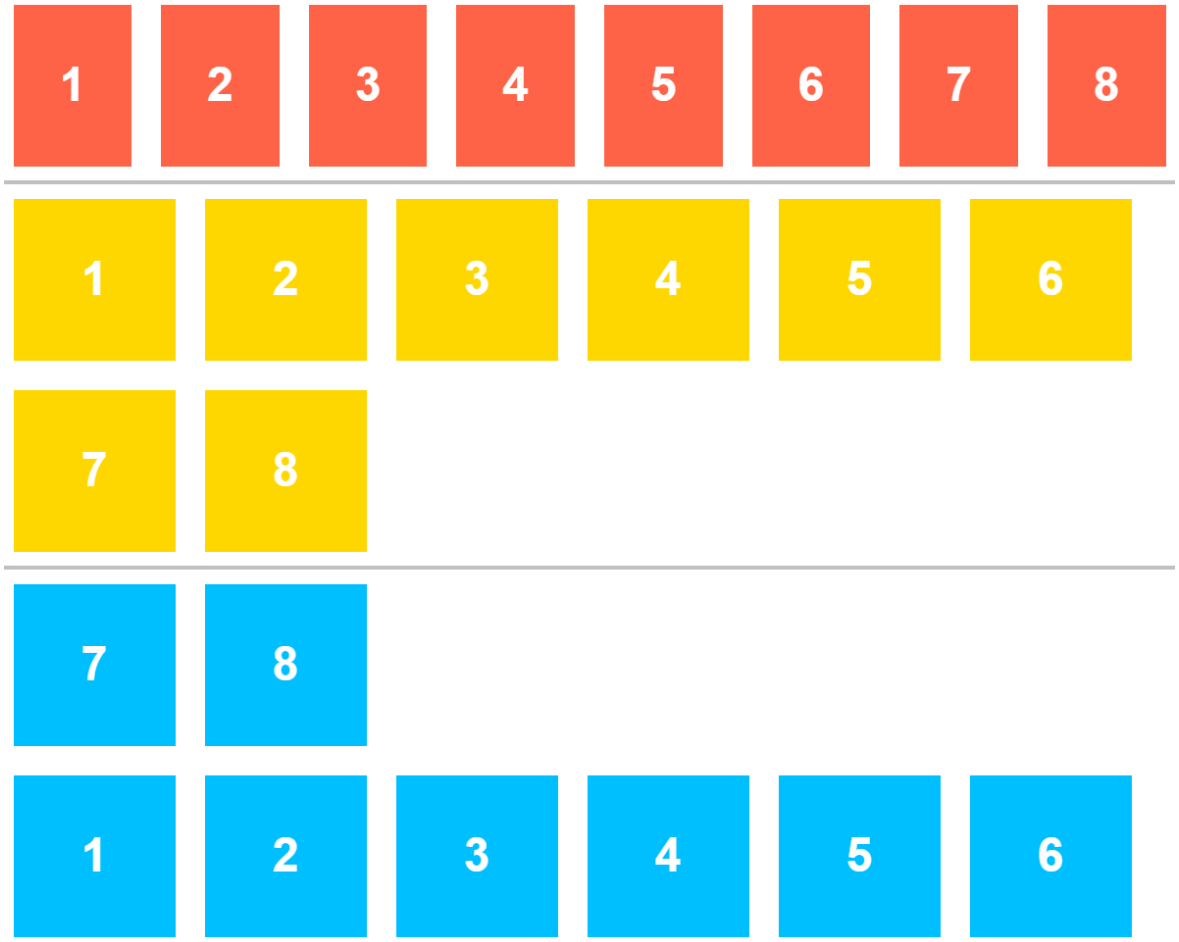
```
<!DOCTYPE html>
<html>
<head>
</head>
<body>

<div class="flex-container">
  <div class="flex-item">
        flex item 1
  </div>
  <div class="flex-item">
        flex item 2
  </div>
  <div class="flex-item">
        flex item 3
  </div>
</div>

</body>
</html>
```

```
body {
    font-family: arial;
    color: white;
}

.flex-container {
    display: flex;
    align-items: center;
    width: 50%;
    height: 250px;
    background-color: grey;
}

.flex-item {
    background-color: darkred;
    width: 20%;
    height: 100px;
    margin: 10px;
    padding: 10px;
}
```

Flex-wrap

- La proprietà **flex-wrap** specifica se il ***flex items*** debba andare automaticamente a capo oppure no, nel caso in cui non ci sia spazio sufficiente su un'unica ***flex line***
- Il valore di default è **nowrap**: e corrisponde al non andare a capo automaticamente, comprimendo lo spazio
- Altri possibili valori sono:
  - **Wrap**: I flex item vengono mandati a capo automaticamente, se necessario (nell'esempio)
  - **wrap-reverse:** I flex item vanno a capo automaticamente, se necessario, ma in ordine inverso

Flex-wrap



flex_wrap_06.html

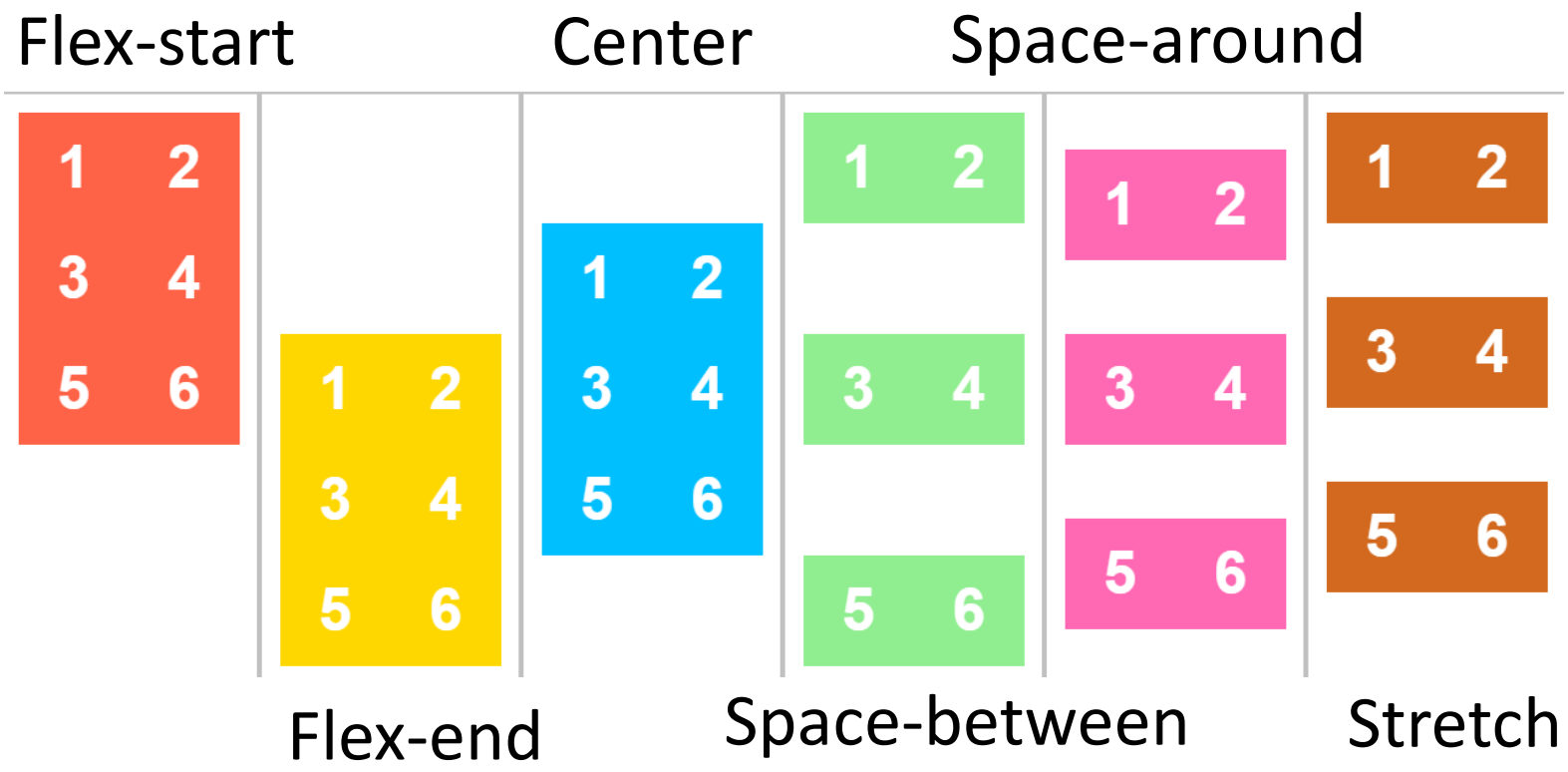
```
<!DOCTYPE html>
<html>
<head>
</head>
<body>

<div class="flex-container">
  <div class="flex-item">
        flex item 1
  </div>
  <div class="flex-item">
        flex item 2
  </div>
  <div class="flex-item">
        flex item 3
  </div>
</div>

</body>
</html>
```

```
body {
    font-family: arial;
    color: white;
}

.flex-container {
    display: flex;
    flex-wrap: wrap;
    width: 450px;
    height: 400px;
    background-color: grey;
}

.flex-item {
    background-color: darkred;
    width: 150px;
    height: 100px;
    margin: 10px;
    padding: 10px;
}
```

Align-content

- La proprietà **align-content** modifica il comportamento della proprietà **flex-wrap**
- E' simile a **align-items**, ma invece di allineare i ***flex item***, allinea le ***flex line***
- Il valore di default è **stretch**: le linee sono "stretchate" per occupare lo spazio restante
- Gli altri possibili valori sono:
  - **flex-start**
  - **flex-end**
  - **center** (`align_content_07.html`)
  - **space-between**
  - **space-around**

Align-content



align_content_07.html

```
<!DOCTYPE html>
<html>
<head>
</head>
<body>

<div class="flex-container">
  <div class="flex-item">
        flex item 1
  </div>
  <div class="flex-item">
        flex item 2
  </div>
  <div class="flex-item">
        flex item 3
  </div>
</div>

</body>
</html>
```

```
body {
    font-family: arial;
    color: white;
}
.flex-container {
    display: flex;
    flex-wrap: wrap;
    align-content: center;
    width: 450px;
    height: 400px;
    background-color: grey;
}
.flex-item {
    background-color: darkred;
    width: 150px;
    height: 100px;
    margin: 10px;
    padding: 10px;
}
```

Order

- La proprietà **order** specifica l'ordine di un ***flex item*** relativo al resto degli altri ***flex item*** all'interno dello stesso ***container***
- Valori ammessi: interi (positivi e negativi)
- Il prossimo esempio mostra il risultato dell'uso di **order** in un ***flex item***:
  `order_08.html`

order_08.html

```
<!DOCTYPE html>
<html>
<head>
</head>
<body>

<div class="flex-container">
  <div class="flex-item">
        flex item 1
  </div>
  <div class="flex-item first">
        flex item 2
  </div>
  <div class="flex-item">
        flex item 3
  </div>
</div>

</body>
</html>
```

```
body {
    font-family: arial;
    color: white;
}
.flex-container {
    display: flex;
    width: 400px;
    height: 250px;
    background-color: grey;
}
.flex-item {
    background-color: darkred;
    width: 100px;
    height: 100px;
    margin: 10px;
    padding: 10px;
}
.first {
    order: -1;
}
```

Margin

- **`margin: auto;`**
  assorbe lo spazio extra. Può essere usato per spingere i ***flex item*** verso posizioni diverse
- Nel prossimo esempio impostiamo

`margin-right: auto;`

sul primo ***flex item***.
Questo causerà l'assorbimento dello spazio extra alla destra dell'elemento:
`margin_09.html`

margin_09.html

```
<!DOCTYPE html>
<html>
<head>
</head>
<body>

<div class="flex-container">
  <div class="flex-item right">
        flex item 1
  </div>
  <div class="flex-item">
        flex item 2
  </div>
  <div class="flex-item">
        flex item 3
  </div>
</div>

</body>
</html>
```

```
body {
    font-family: arial;
    color: white;
}
.flex-container {
    display: flex;
    width: 500px;
    height: 250px;
    background-color: grey;
}
.flex-item {
    background-color: darkred;
    width: 100px;
    height: 100px;
    margin: 10px;
    padding: 10px;
}
.right {
    margin: auto;
}
```

Perfect Centering

- Nel prossimo esempio vedremo come posizionare un elemento perfettamente al centro del suo elemento contenitore
- Questa soluzione è nota come: *perfect centering*
- Finalmente con i flexbox è possibile e semplice:

```
margin: auto;
```

- Questa regola permette di centrare perfettamente un elemento rispetto ad entrambi gli assi (**`centering_10.html`**)

centering_10.html

```
<!DOCTYPE html>
<html>
<head>
</head>
<body>

<div class="flex-container">
  <div class="flex-item">
   flex item (perfect centering)
  </div>
</div>

</body>
</html>
```

```
body {
    font-family: arial;
    color: white;
}
.flex-container {
    display: flex;
    width: 400px;
    height: 250px;
    background-color: grey;
}
.flex-item {
    background-color: darkred;
    width: 100px;
    height: 100px;
    margin: auto;
    padding: 10px;
}
```

Align-self

- La proprietà **align-self** di un **flex item** sovrascrive la proprietà **align-items** del **flex container** per quell particolare ***item***
- Ammette gli stessi possibili valori della proprietà **align-items** (**stretch**, **flex-start**, **flex-end**, **center**, **baseline**)
- Il prossimo esempio mostra il risultato di differenti valori di **align-self** assegnati ad ogni flex item (**`align_self_11.html`**)

align_self_11.html

```
<!DOCTYPE html>
<html>
<head>
</head>
<body>

<div class="flex-container">
  <div class="flex-item item1">flex-start</div>
  <div class="flex-item item2">flex-end</div>
  <div class="flex-item item3">center</div>
  <div class="flex-item item4">baseline</div>
  <div class="flex-item item5">stretch</div>
</div>

</body>
</html>
```

align_self_11.html

```
body {
    font-family: arial;
    color: white;
}
.flex-container {
    display: flex;
    width: 600px;
    height: 250px;
    background-color: grey;
}
.flex-item {
    background-color: darkred;
    width: 100px;
    min-height: 100px;
    margin: 10px;
    padding: 10px;
    text-align: center;
}
```

```
.item1 {
    align-self: flex-start;
}
.item2 {
    align-self: flex-end;
}
.item3 {
    align-self: center;
}
.item4 {
    align-self: baseline;
}
.item5 {
    align-self: stretch;
}
```

Flex-grow

- La proprietà **flex-grow** definisce la capacità di un elemento di crescere, specificando la ***lunghezza*** del **flex item** relativa agli altri ***flex item*** all'interno dello stesso ***container***
- Si può specificare un valore unitario per alcuni elementi e un valore più elevato per altri
- Così se su tre elementi due hanno valore unitario 1 e uno ha valore 2, quest'ultimo cercherà di occupare il doppio dello spazio rispetto agli altri elementi
- Nel prossimo esempio, il primo ***flex item*** occuperà la metà dello spazio libero, mentre gli altri due ***flex item*** occuperanno 1/4 dello spazio libero ciascuno: **`flex_grow_12.html`**

flex_grow_12.html

```
<!DOCTYPE html>
<html>
<head>
</head>
<body>

<div class="flex-container">
    <div class="flex-item double">
     flex item 1
    </div>
    <div class="flex-item single">
     flex item 2
    </div>
    <div class="flex-item single">
     flex item 3
    </div>
</div>

</body>
</html>
```

```
body {
    font-family: arial;
    color: white;}
.flex-container {
    display: flex;
    width: 600px;
    height: 250px;
    background-color: grey;}
.flex-item {
    background-color: darkred;
    margin: 10px;
    text-align: center;}
.double {
    flex-grow: 2;}
.single {
    flex-grow: 1;}
```

Flex-shrink

- La proprietà **flex-shrink** definisce l'abilità di un elemento di contrarsi se necessario.
- Si applica ai **flex item**, ed è relativa agli altri ***flex item*** all'interno dello stesso ***container***
- Nel prossimo esempio, il secondo ***flex item*** occuperà la metà della larghezza occupata di default dagli altri ***flex item***:
  `flex_shrink_13.html`

Flex_shrink_13.html

```
<!DOCTYPE html>
<html>
<head>
</head>
<body>

<div class="flex-container">
    <div class="flex-item">
     flex item 1
    </div>
    <div class="flex-item shrink">
     flex item 2
    </div>
    <div class="flex-item">
     flex item 3
    </div>
</div>

</body>
</html>
```

```
body {
    font-family: arial;
    color: white;}
.flex-container {
    display: flex;
    width: 600px;
    height: 250px;
    background-color: grey;}
.flex-item {
    background-color: darkred;
    margin: 10px;
    text-align: center;
    width: 300px;
}
.shrink {
    flex-shrink: 3;}
```

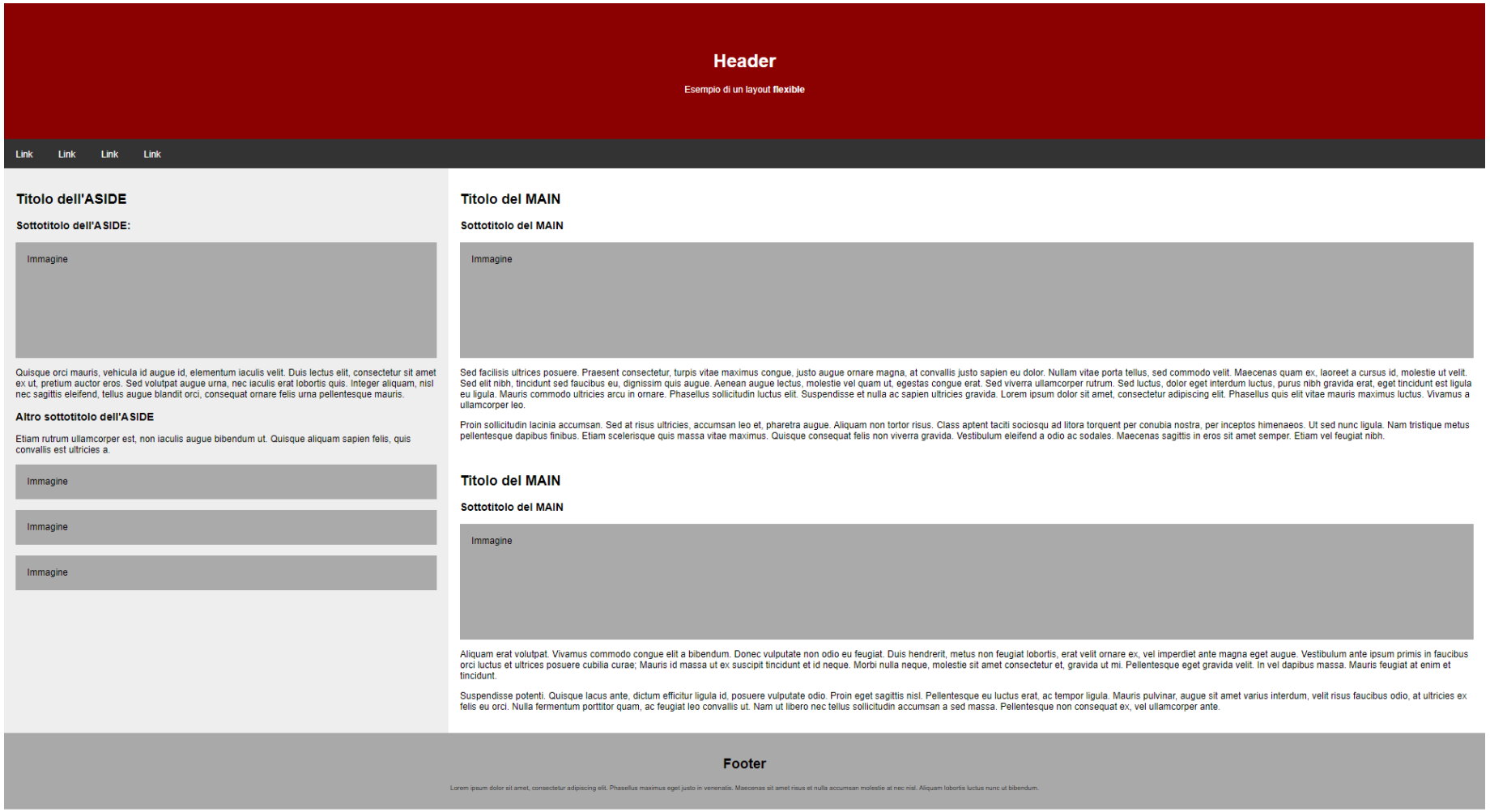
Responsive flexbox

- Questo è un esempio di un layout responsive in cui vogliamo usare i flexbox per gli elementi:
  - **`<header>`**
  - **`<main>`**
  - **`<nav>`**
  - **`<aside>`**
  - **`<footer>`**
- **`Esempio.html`**

No CSS


Header
Esempio di un layout flexible
Link
Link
Link
Link
Titolo dell'ASIDE
Sottotitolo dell'ASIDE:
Immagine
Quisque orci mauris, vehicula id augue id, elementum iaculis velit. Duis lectus elit, consectetur sit amet ex ut, pretium auctor eros. Sed volutpat augue urna, nec iaculis erat lobortis quis. Integer aliquam, nisl nec sagittis eleifend, tellus augue blandit orci, consequat ornare felis urna pellentesque mauris.
Altro sottotitolo dell'ASIDE
Etiam rutrum ullamcorper est, non iaculis augue bibendum ut. Quisque aliquam sapien felis, quis convallis est ultricies a.
Immagine
Immagine
Immagine
Titolo del MAIN
Sottotitolo del MAIN
Immagine
Sed facilisis ultrices posuere. Praesent consectetur, turpis vitae maximus congue, justo augue ornare magna, at convallis justo sapien eu dolor. Nullam vitae porta tellus, sed commodo velit. Maecenas quam ex, laoreet a cursus id, molestie ut velit. Sed elit nibh, tincidunt sed faucibus eu, dignissim quis augue. Aenean augue lectus, molestie vel quam ut, egestas congue erat. Sed viverra ullamcorper rutrum. Sed luctus, dolor eget interdum luctus, purus nibh gravida erat, eget tincidunt est ligula eu ligula. Mauris commodo ultricies arcu in ornare. Phasellus sollicitudin luctus elit. Suspendisse et nulla ac sapien ultricies gravida. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Phasellus quis elit vitae mauris maximus luctus. Vivamus a ullamcorper leo.
Proin sollicitudin lacinia accumsan. Sed at risus ultricies, accumsan leo et, pharetra augue. Aliquam non tortor risus. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos himenaeos. Ut sed nunc ligula. Nan tristique metus pellentesque dapibus finibus. Etiam scelerisque quis massa vitae maximus. Quisque consequat felis non viverra gravida. Vestibulum eleifend a odio ac sodales. Maecenas sagittis in eros sit amet semper. Etiam vel feugiat nibh.
Titolo del MAIN
Sottotitolo del MAIN
Immagine
Aliquam erat volutpat. Vivamus commodo congue elit a bibendum. Donec vulputate non odio eu feugiat. Duis hendrerit, metus non feugiat lobortis, erat velit ornare ex, vel imperdiet ante magna eget augue. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia curae; Mauris id massa ut ex suscipit tincidunt et id neque. Morbi nulla neque, molestie sit amet consectetur et, gravida ut mi. Pellentesque eget gravida velit. In vel dapibus massa. Mauris feugiat at enim et tincidunt.
Suspendisse potenti. Quisque lacus ante, dictum efficitur ligula id, posuere vulputate odio. Proin eget sagittis nisl. Pellentesque eu luctus erat, ac tempor ligula. Mauris pulvinar, augue sit amet varius interdum, velit risus faucibus odio, at ultricies ex felis eu orci. Nulla fermentum porttitor quam, ac feugiat leo convallis ut. Nam ut libero nec tellus sollicitudin accumsan a sed massa. Pellentesque non consequat ex, vel ullamcorper ante.
Footer
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Phasellus maximus eget justo in venenatis. Maecenas sit amet risus et nulla accumsan molestie at nec nisl. Aliquam lobortis luctus nunc ut bibendum.
Header
Nav
Aside
Main
Footer

Risultato che vogliamo ottenere



Responsive flexbox

- Lavoriamo insieme aggiungendo il foglio di stile interno per ottenere il risultato mostrato nella slide precedente e che troverete nel file **`esempio_completo.html`** su Virtuale

Riferimenti

- Risorse online sulla piattaforma
- Standard completi:
  - CSS1, http://www.w3.org/TR/CSS1
  - CSS2, http://www.w3.org/TR/CSS2
- https://www.w3.org/TR/css-flexbox-1/
- Approfondimenti su «CSS3 Guida completa per lo sviluppatore», Peter Gasston, 2011. Disponibile in biblioteca

Risorse online

- http://www.cssauthor.com/css-flexbox/
- https://css-tricks.com/snippets/css/a-guide-to-flexbox/
- http://learnlayout.com/flexbox.html

- Edutainment:
  - http://flexboxfroggy.com/
  - http://www.flexboxdefense.com/