CSS Layout

Arrangement and Positioning of the HTML Elements

Au Mau Duong
Technical Trainers

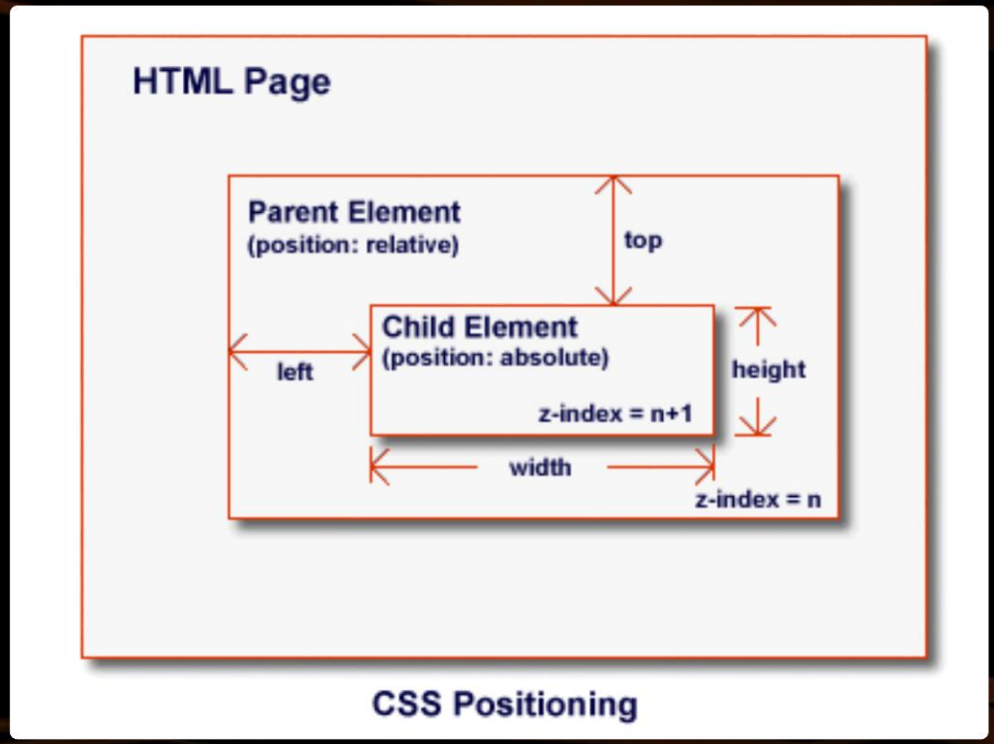
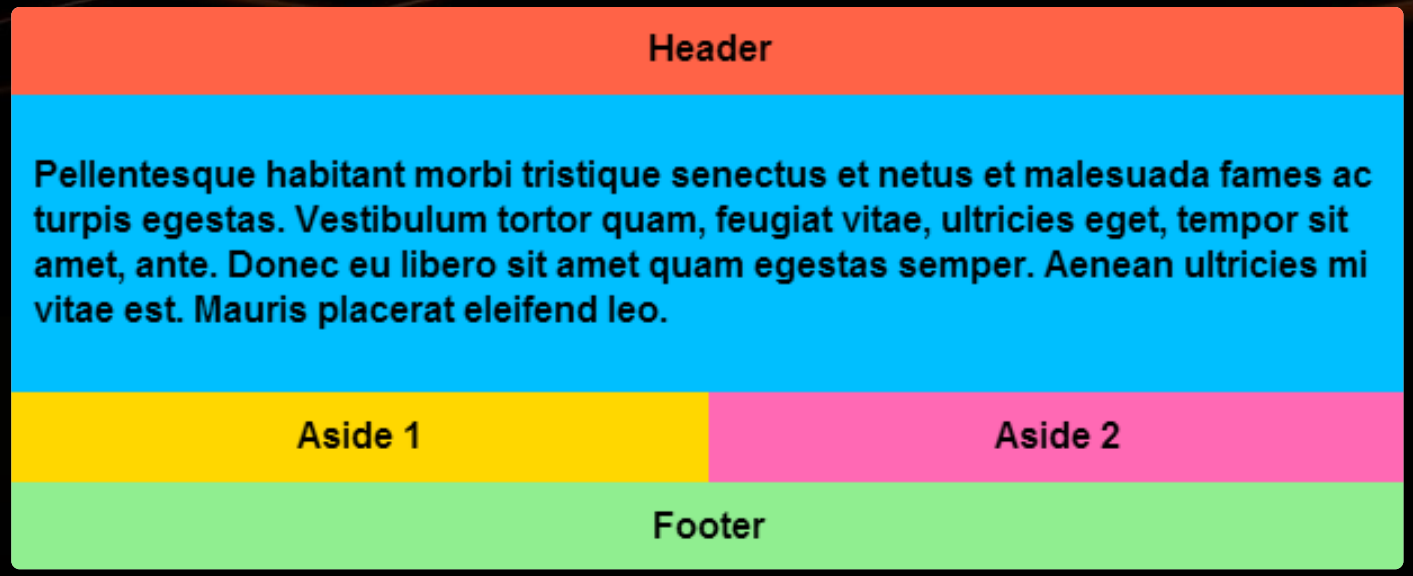


Table of Contents

- Flexbox
  - flex-direction, flex-wrap, flex-flow, container properties
  - justify-content, align-content, align-items
- Transforms
  - CSS 2D Transforms
  - CSS 3D Transforms

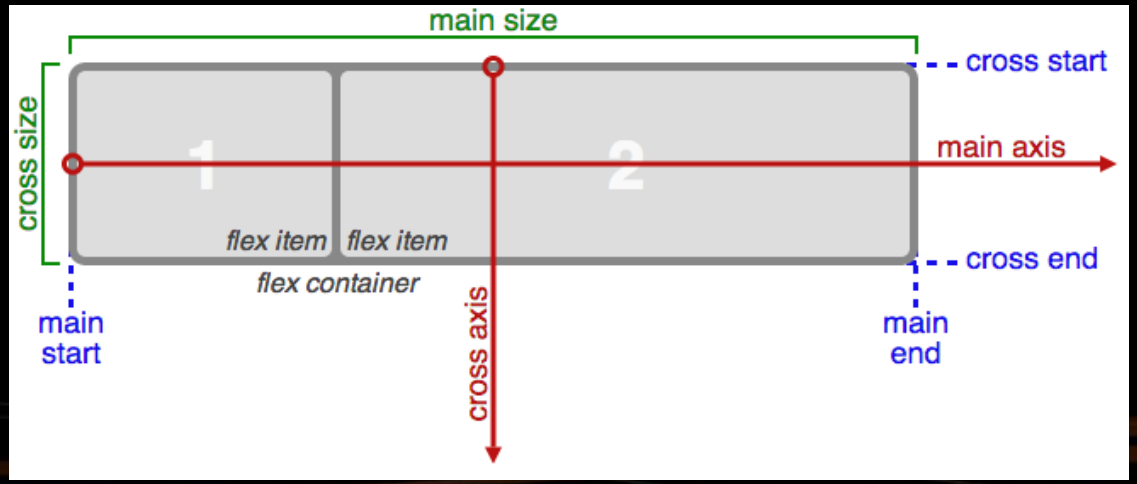


Flexbox

The Magic of `display:flex`

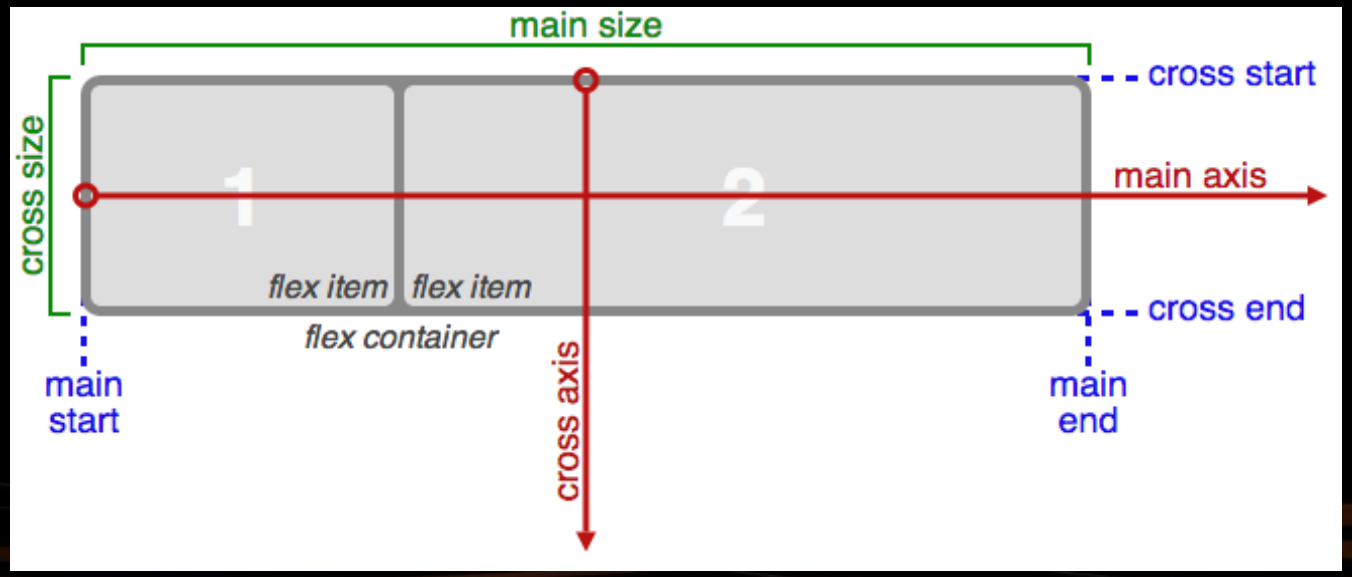
Flexbox Basics (1)

- **Flexbox (Flexible layout) -** efficient way to lay out, align and distribute space among items in a container
  - Gives the container ability to alter its items' width/height/order to best fill the available space
- **main axis** - primary axis along which flex items are laid out
  - Horizontal or vertical - depends on flex-direction property
- **main size** - flex items' width or height, whichever is in the main dimension
- **main-start / main-end** - flex items are placed within the container starting from main-start and going to main-end



Flexbox Basics (2)

- **cross axis** - perpendicular to the main axis
- **cross size** - the width or height of a flex item, whichever is in the cross dimension
- **cross-start / cross-end** - Flex lines are filled with items and placed into the container starting on the cross-start side of the flex container and going toward the cross-end side.



Flex - container properties (1)

- **display: flex** - defines a flex container
- **flex-direction** - primarily laying out either in horizontal rows or vertical columns
  - `row`: left to right (`ltr`) right to left (`rtl`)
  - `row-reverse`: right to left (`ltr`) left to right (`rtl`)
  - `column`: same as row, but top to bottom
  - `column-reverse`: same as row-reverse but bottom to top
- **flex-wrap** - wrap as needed with this property
  - `Nowrap`: single-line / left to right (`ltr`) right to left (`rtl`)
  - `Wrap`: multi-line / left to right (`ltr`) right to left (`rtl`)
  - `Wrap-reverse`: multi-line / right to left (`ltr`) left to right (`rtl`)
- **flex-flow** - shorthand for (`flex-direction`) & (`flex-wrap`) properties

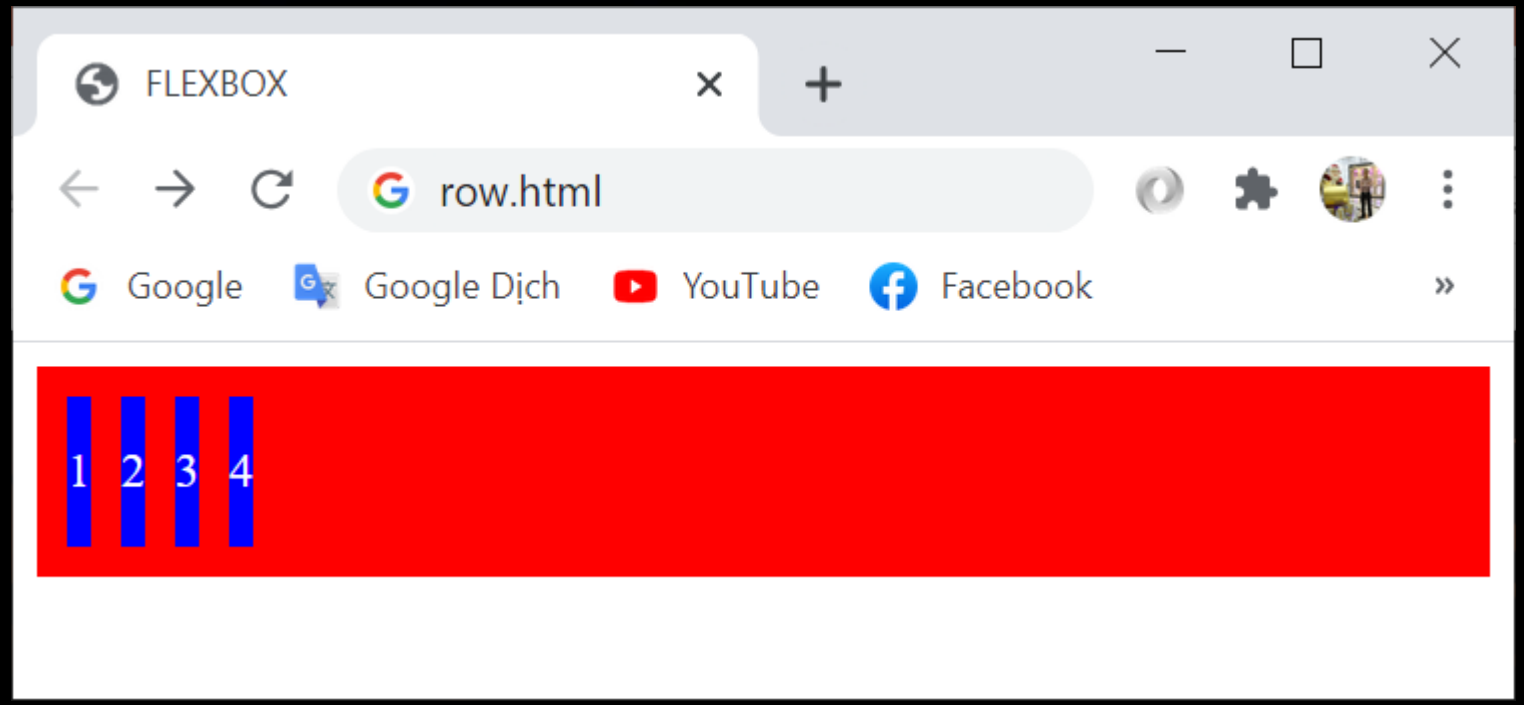
flex-direction Demo

```
<div class="container">
    <div class="item item1">1</div>
    <div class="item item2">2</div>
    <div class="item item3">3</div>
    <div class="item item4">4</div>
</div>
```

```
<style>
.container {
 background: red;
 max-width: 960px;
 max-height: 1000px;
 margin: 0 auto;
 padding: 5px;
}
.item {
 background: blue;
 margin: 5px;
 color: white;
 height: 50px;
 text-align: center;
 line-height: 50px;
}
</style>
```

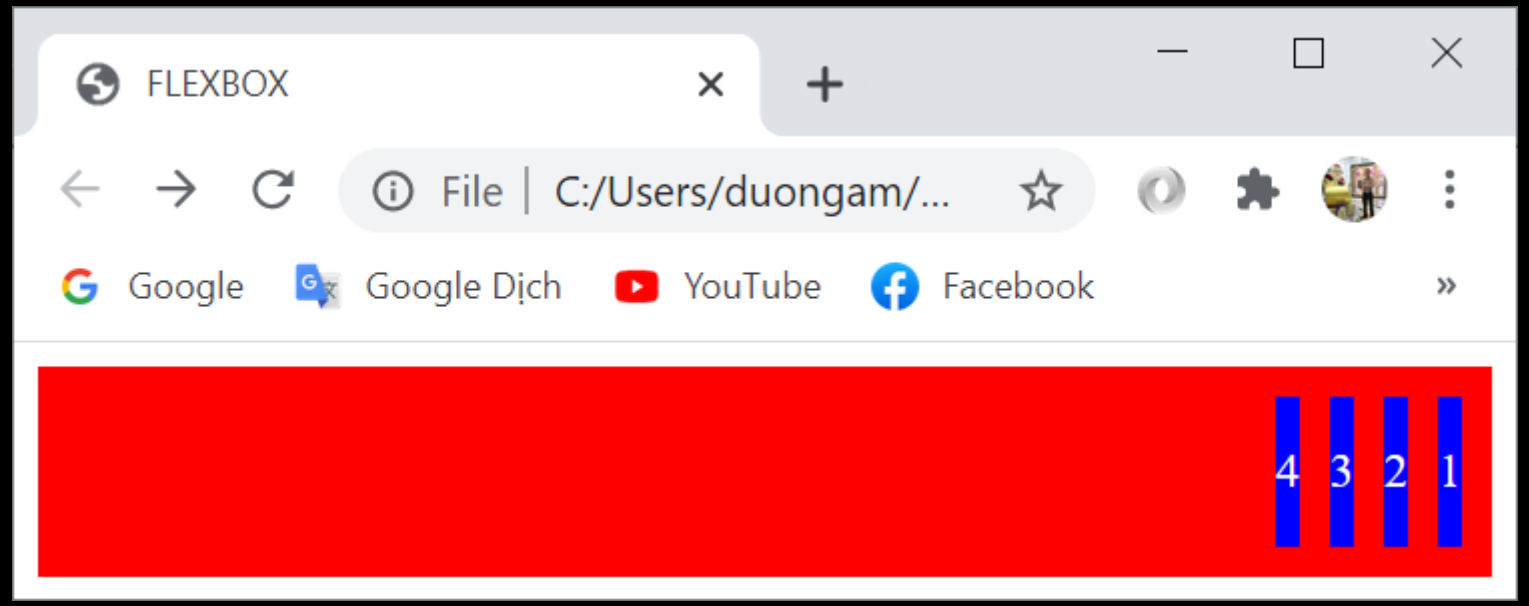
flex-direction: row

```
<style>
.container {
 display: flex;
 flex-direction: row
}
</style>
```



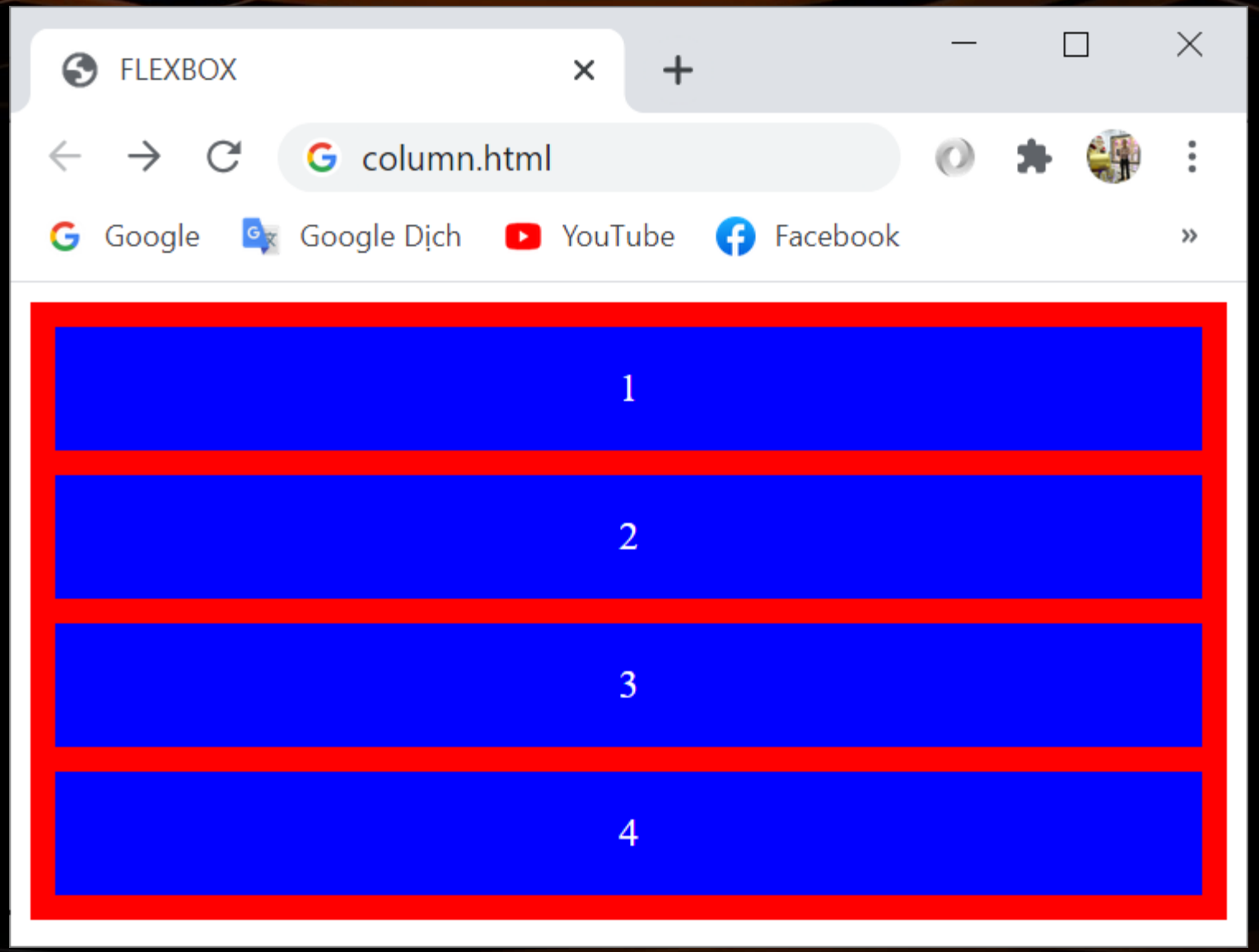
flex-direction: row-reverse

```
<style>
.container {
 display: flex;
 flex-direction: row-reverse
}
</style>
```



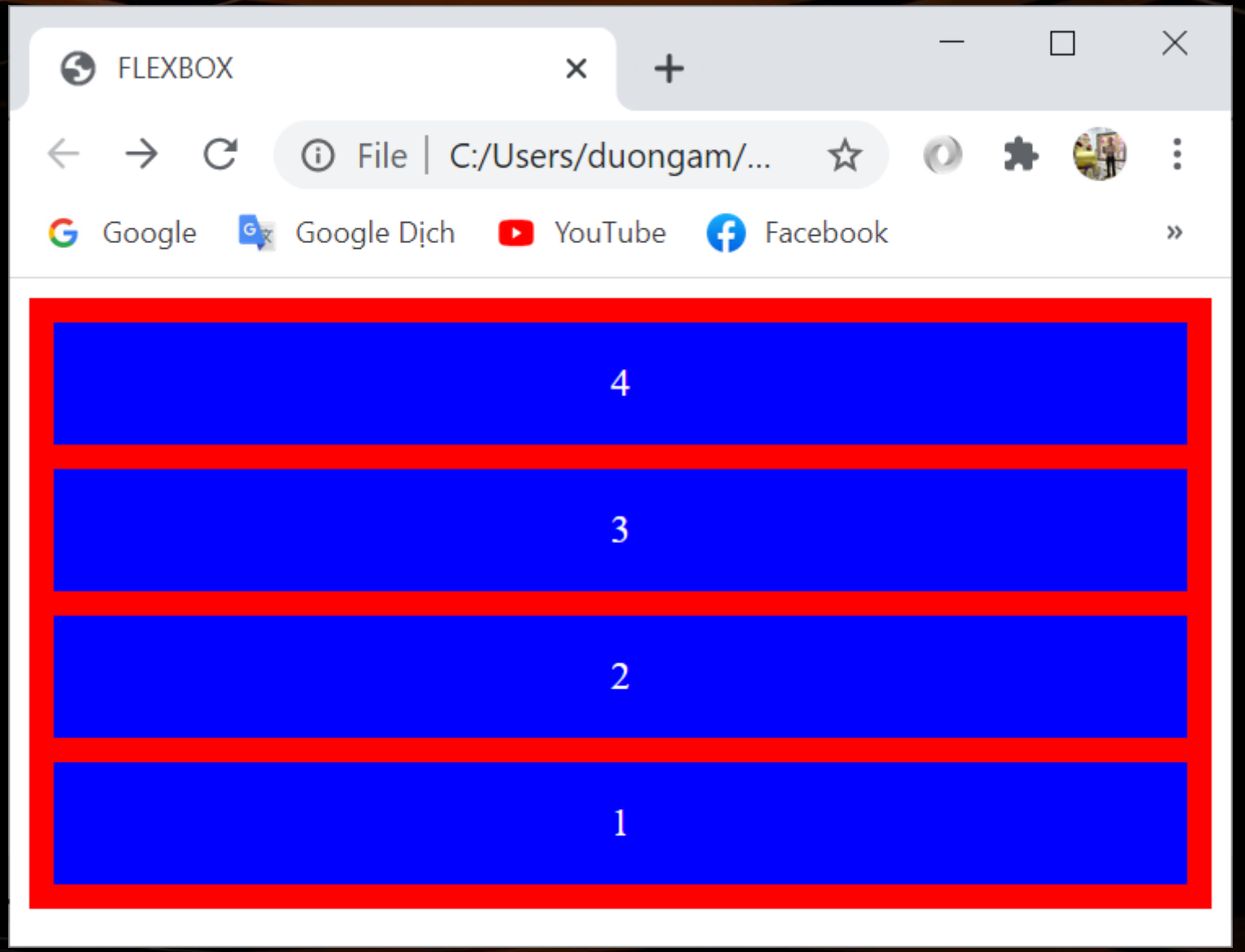
flex-direction: column

```
<style>
.container {
 display: flex;
 flex-direction: column
}
</style>
```



flex-direction: column-reverse

```
<style>
.container {
 display: flex;
 flex-direction:
column-reverse
}
</style>
```

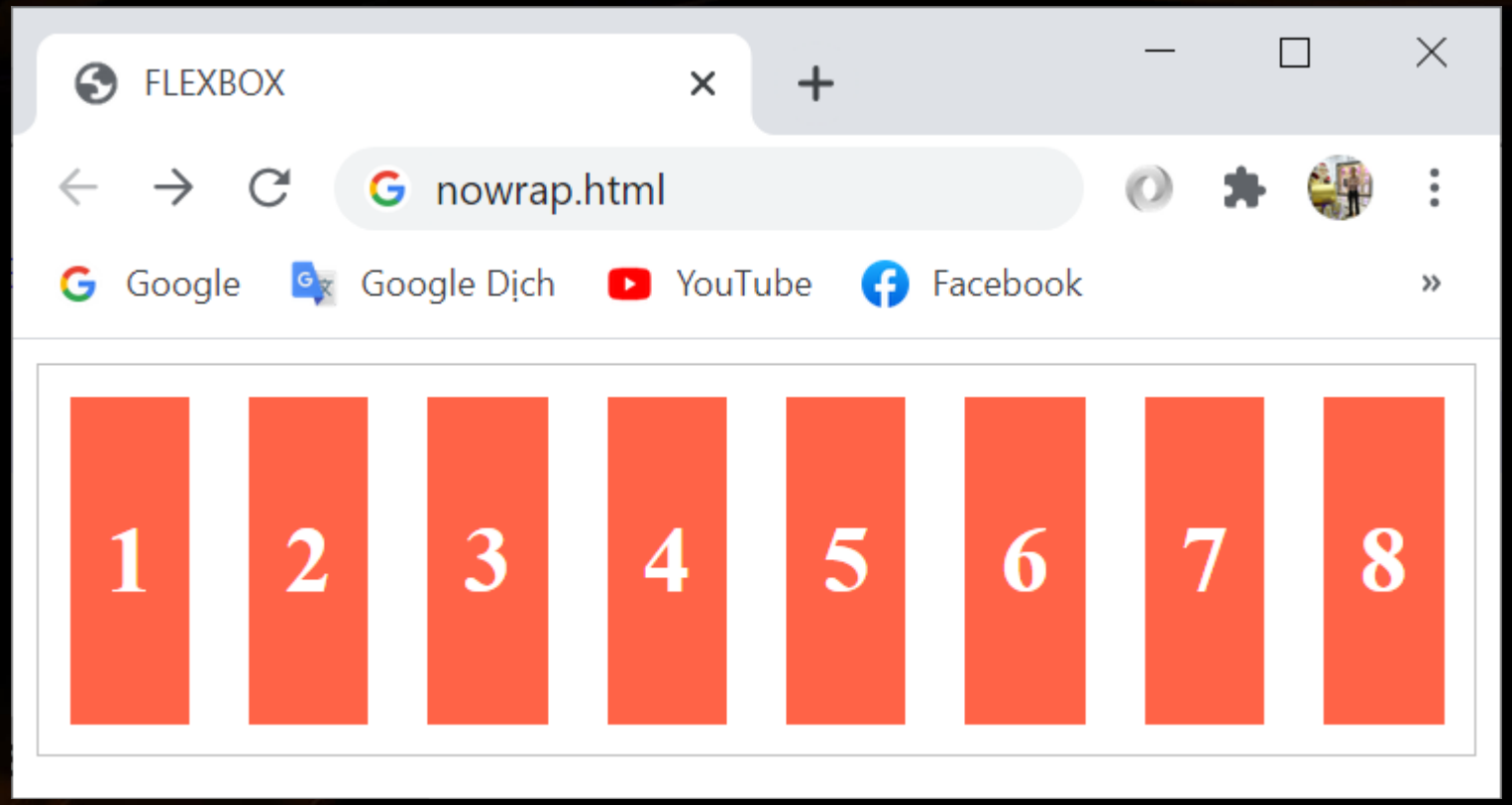


flex-wrap Demo

```
<ul class="flex-container">
  <li class="flex-item">1</li>
  <li class="flex-item">2</li>
  <li class="flex-item">3</li>
  <li class="flex-item">4</li>
  <li class="flex-item">5</li>
  <li class="flex-item">6</li>
  <li class="flex-item">7</li>
  <li class="flex-item">8</li>
</ul>
```

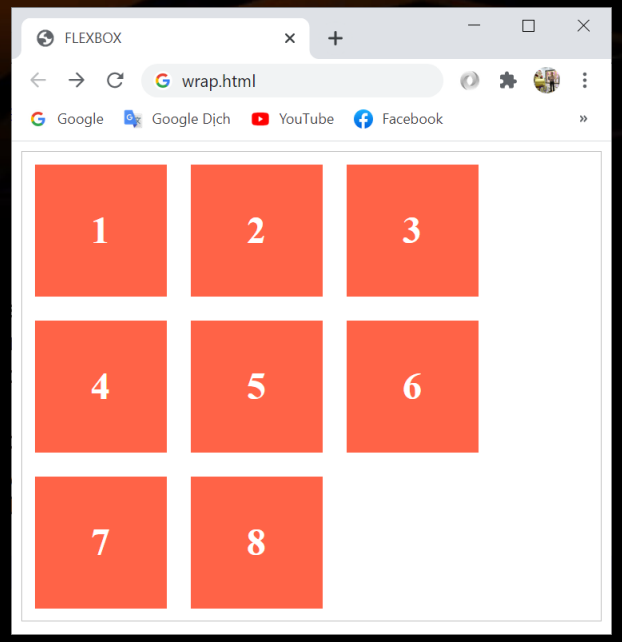
```
<style>
.flex-item {
  background: tomato;
  padding: 5px;
  width: 100px;
  height: 100px;
  margin: 10px;
  line-height: 100px;
  color: white;
  font-weight: bold;
  font-size: 2em;
  text-align: center;
}
</style>
```

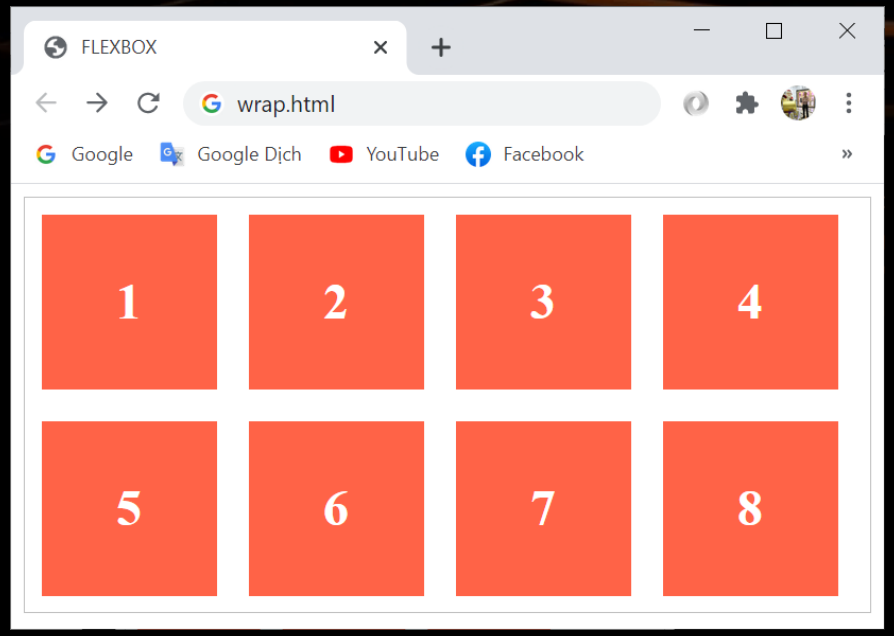
flex-wrap: nowrap

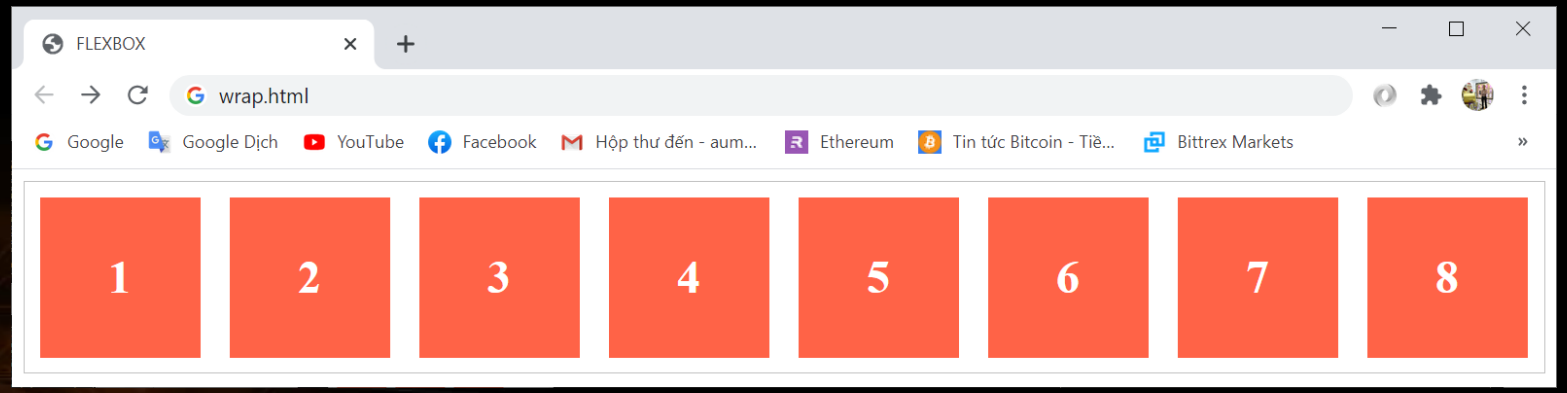



```
<style>
.flex-container {
  padding: 0;
  margin: 0;
  list-style: none;
  border: 1px solid silver;
  -ms-box-orient:
horizontal;
  display: -webkit-box;
  display: -moz-box;
  display: -ms-flexbox;
  display: -moz-flex;
  display: -webkit-flex;
  display: flex;
  flex-wrap: nowrap;
}
</style>
```

flex-wrap: wrap

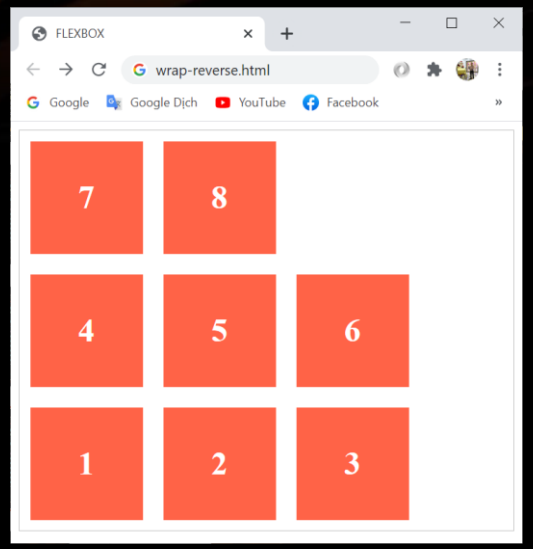


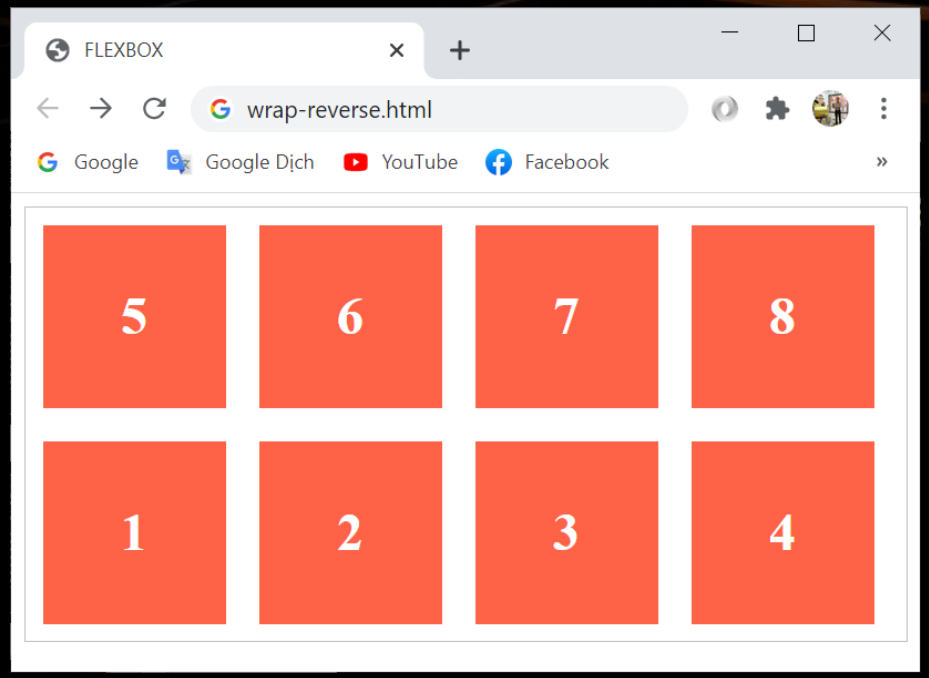


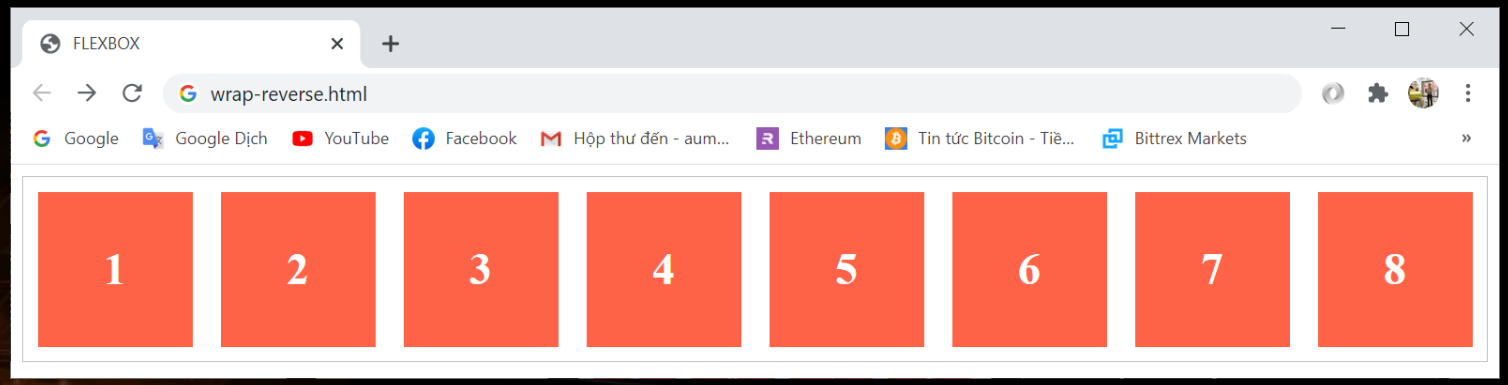



```
<style>
.flex-container {
  padding: 0;
  margin: 0;
  list-style: none;
  border: 1px solid silver;
  -ms-box-orient:
horizontal;
  display: -webkit-box;
  display: -moz-box;
  display: -ms-flexbox;
  display: -moz-flex;
  display: -webkit-flex;
  display: flex;
  flex-wrap: wrap;
}
</style>
```

flex-wrap: wrap-reverse



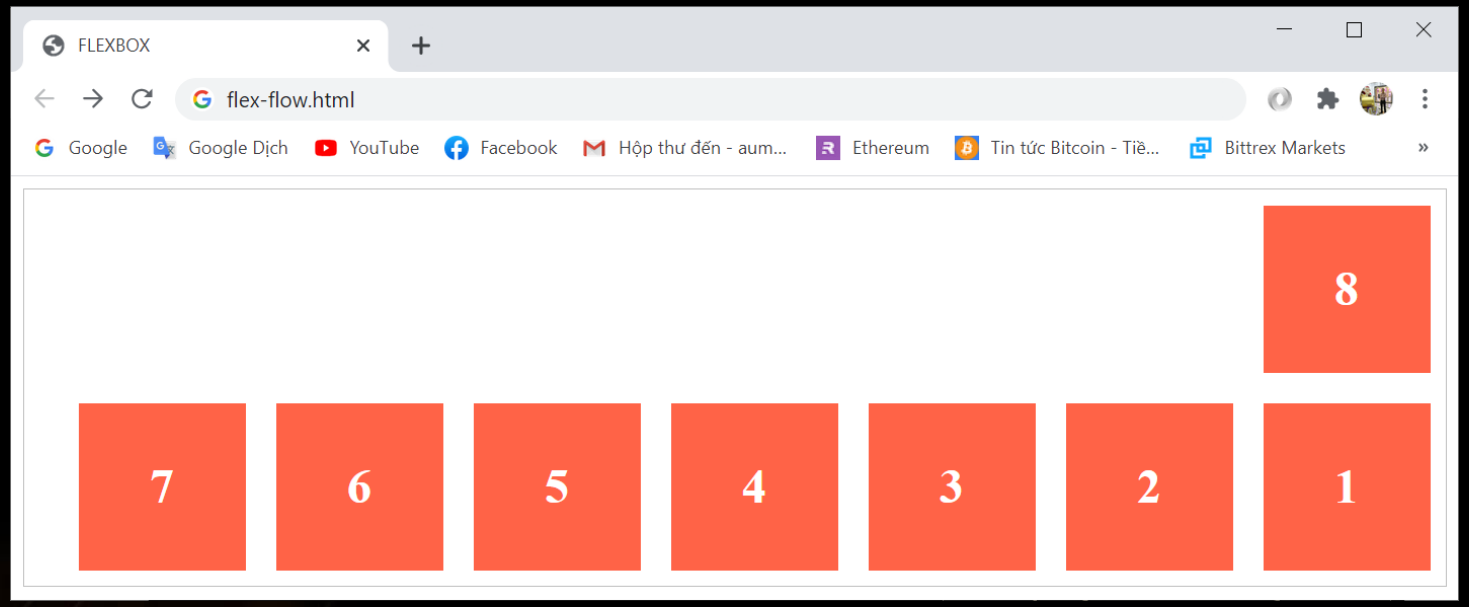





```
<style>
.flex-container {
  padding: 0;
  margin: 0;
  list-style: none;
  border: 1px solid silver;
  -ms-box-orient:
horizontal;
  display: -webkit-box;
  display: -moz-box;
  display: -ms-flexbox;
  display: -moz-flex;
  display: -webkit-flex;
  display: flex;
  flex-wrap: wrap-reverse;
}
</style>
```

flex-flow

shorthand for (**flex-direction**) & (**flex-wrap**) properties



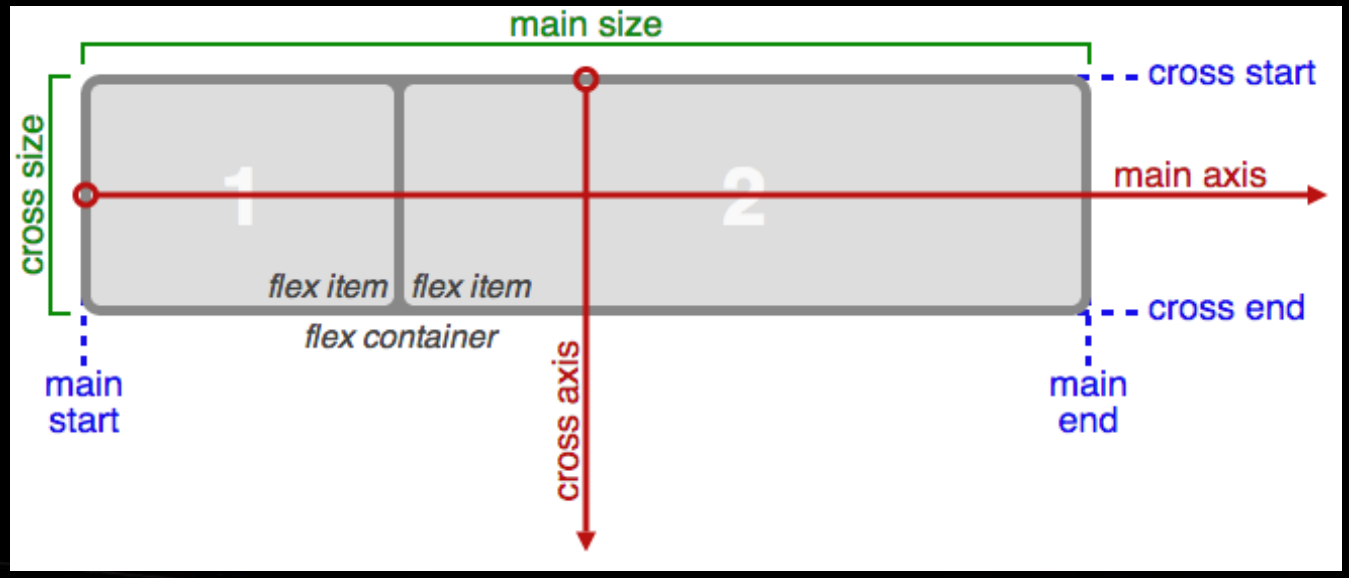

```
<style>
.flex-container {
  padding: 0;
  margin: 0;
  list-style: none;
  border: 1px solid silver;
  -ms-box-orient:
horizontal;
  display: -webkit-box;
  display: -moz-box;
  display: -ms-flexbox;
  display: -moz-flex;
  display: -webkit-flex;
  display: flex;
  flex-flow: row-reverse
wrap-reverse;
}
</style>
```

Flex - container properties (2)

- **justify-content** – alignment along the main axis
- **align-content** – alignment along the cross axis
- **justify-content** and **align-content** property **values**
  - `flex-start`: lines packed to the start of the container
  - `flex-end`: lines packed to the end of the container
  - `center`: lines packed to the center of the container
  - `space-between`: first-to the start & last-to the end of the section
  - `space-around`: equal space between them
- **align-items** – how flex items are laid along the cross axis on the current line
  - `flex-start` | `flex-end` | `center` | `baseline` | `stretch`

Flex - container properties (3)

- **justify-content** – alignment along the main axis
- **align-content** – alignment along the cross axis
- **align-items** – how flex items are laid along the cross axis on the current line



Flex: Demo (1)

```
<style>
.flex-container {
  display: flex;
  height: 600px;
  flex-wrap: wrap;
  background-color: DodgerBlue;
}
```

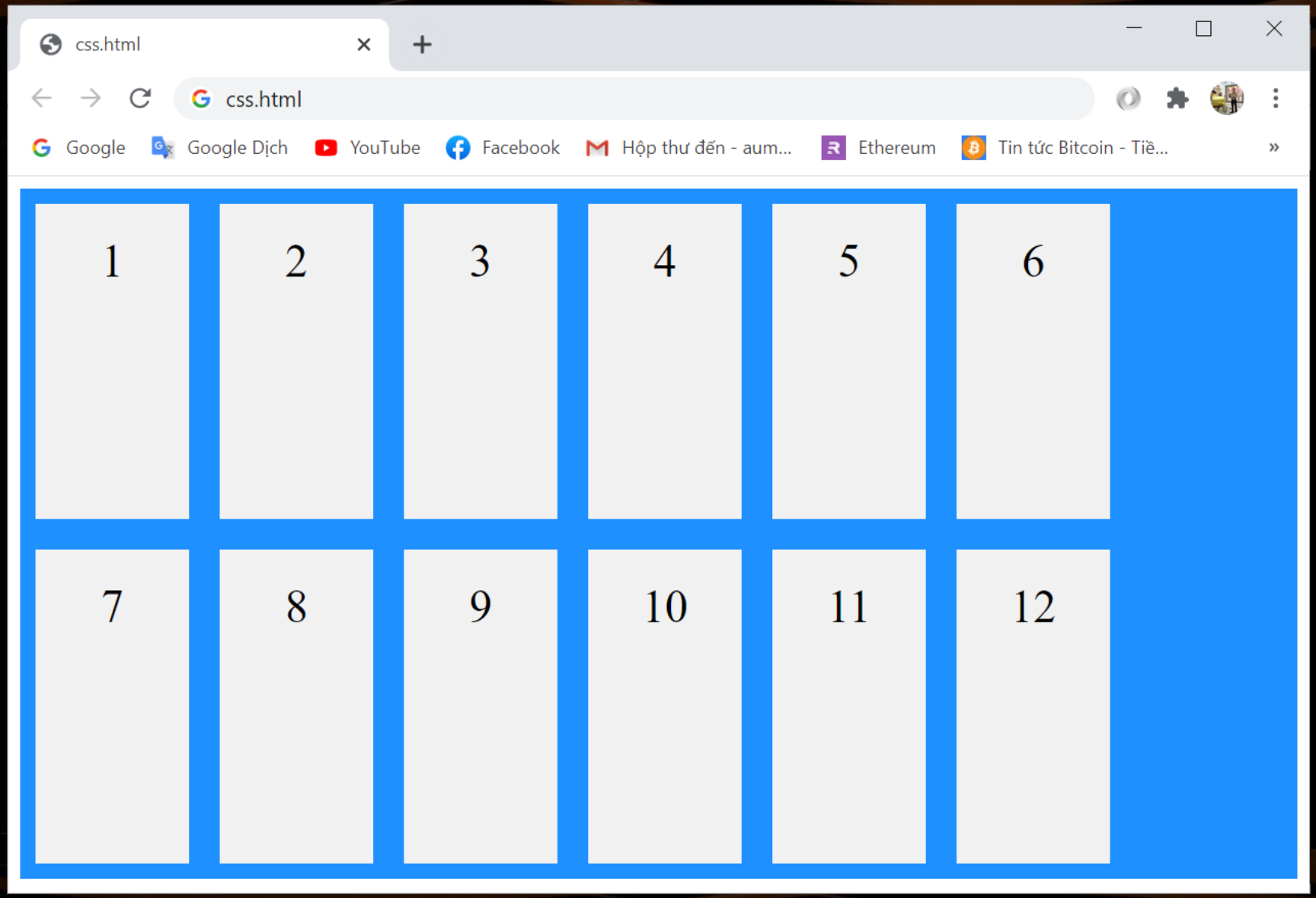
```
.flex-container > div {
  background-color: #f1f1f1;
  width: 100px;
  margin: 10px;
  text-align: center;
  line-height: 75px;
  font-size: 30px;
}
</style>
```

```
<div class="flex-container">
  <div>1</div>
  <div>2</div>
  <div>3</div>
  <div>4</div>
  <div>5</div>
```

```
  <div>6</div>
  <div>7</div>
  <div>8</div>
  <div>9</div>
  <div>10</div>
  <div>11</div>
```

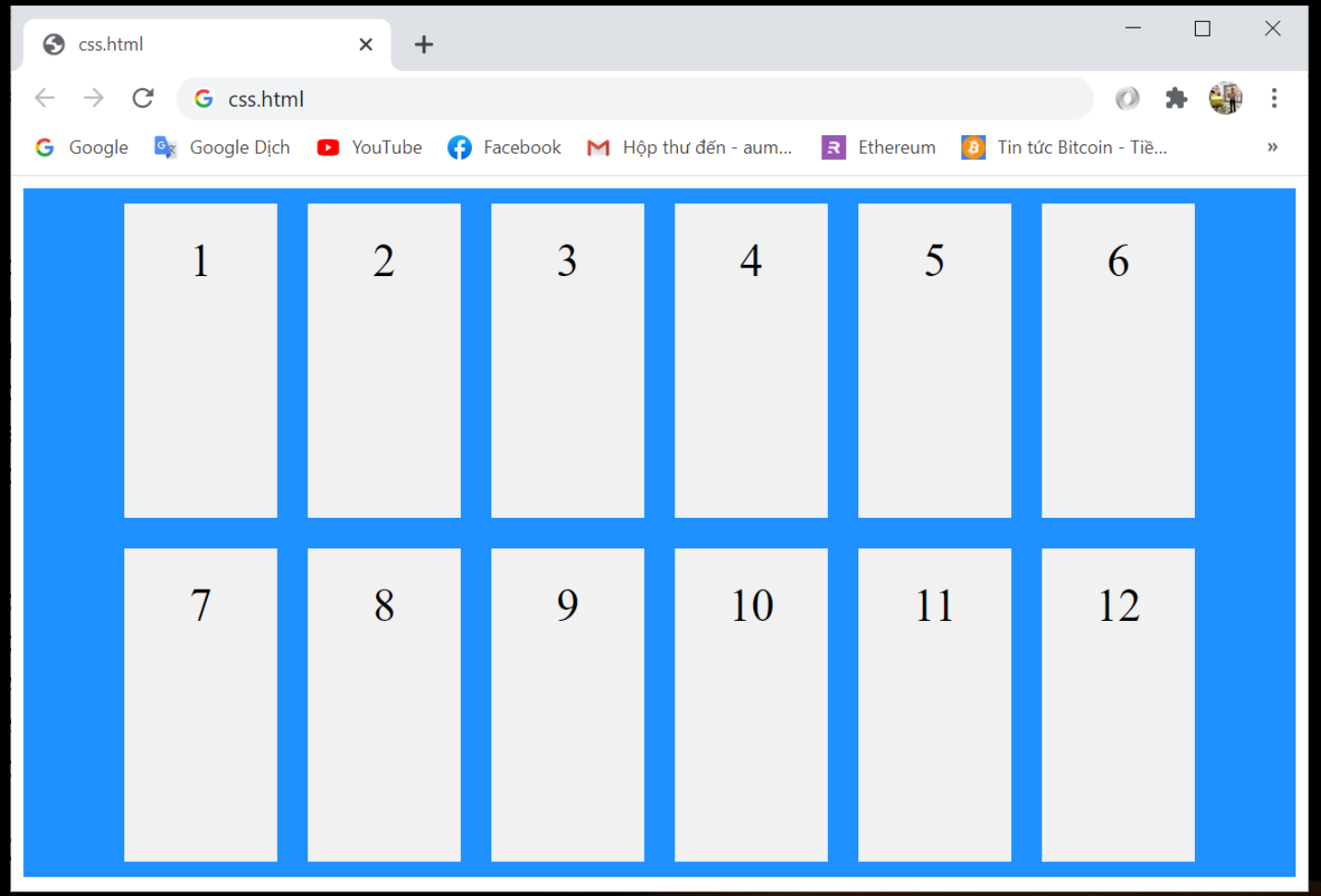
```
<div>12</div>
</div>
```

Flex: Demo (2)


css.html
css.html
Google
Google Dịch
YouTube
Facebook
Hộp thư đến - aum...
Ethereum
Tin tức Bitcoin - Tiề...
1
2
3
4
5
6
7
8
9
10
11
12

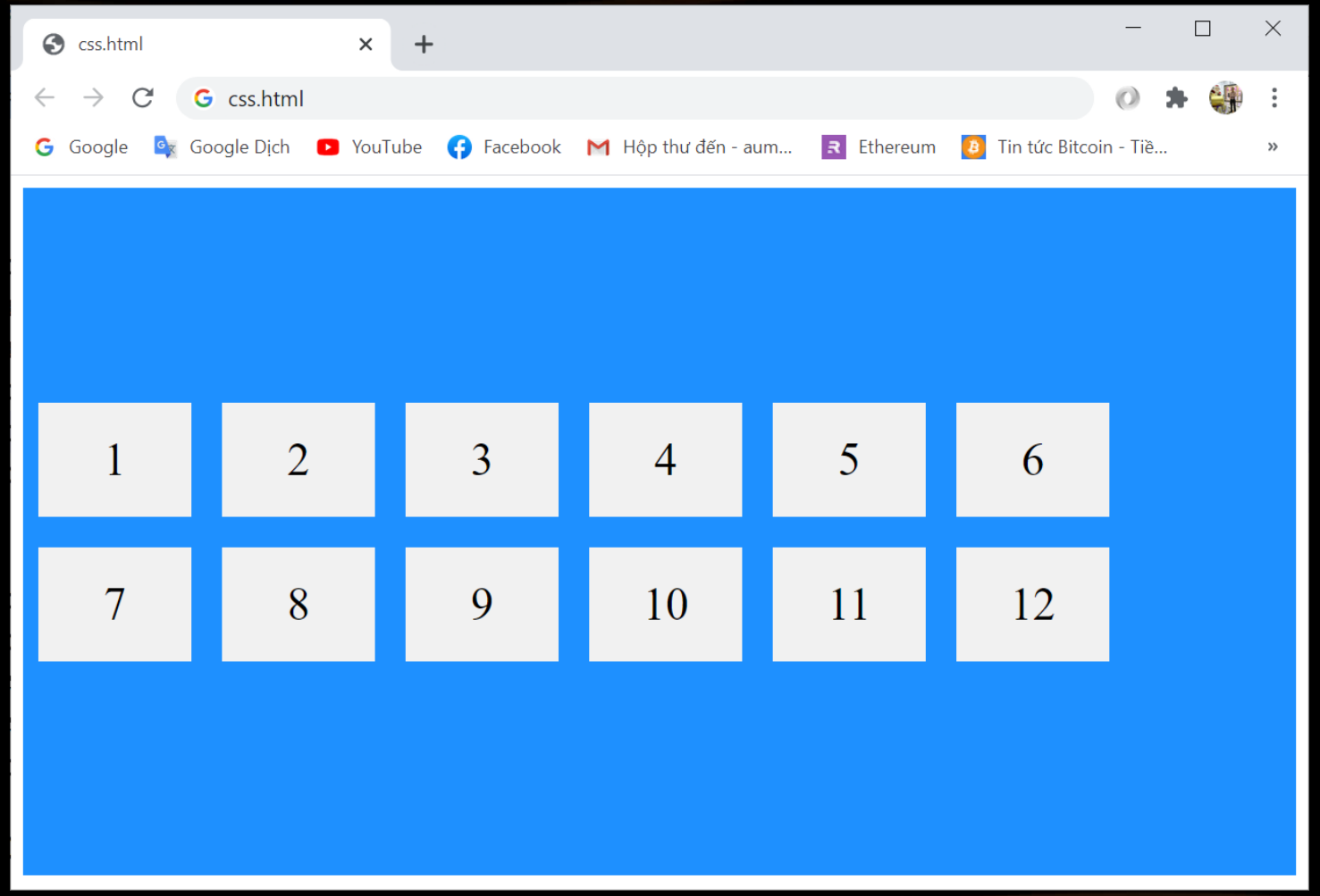
Flex: justify-content

```
<style>
.flex-container {
  justify-content: center;
}
</style>
```



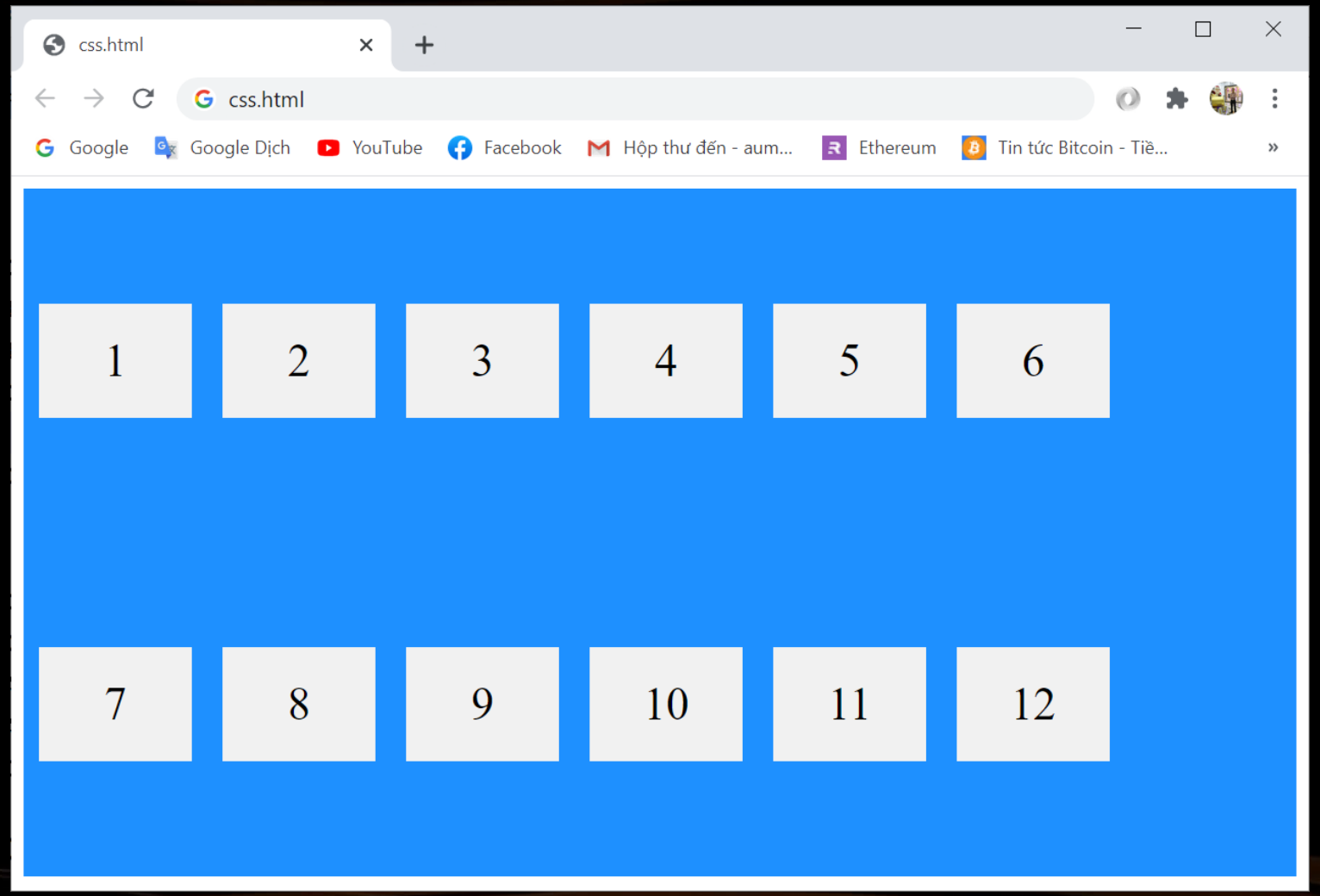
Flex: align-content

```
<style>
.flex-container {
  align-content: center;
}
</style>
```



Flex: align-items

```
<style>
.flex-container {
  align-items: center;
}
</style>
```



Flex – item properties

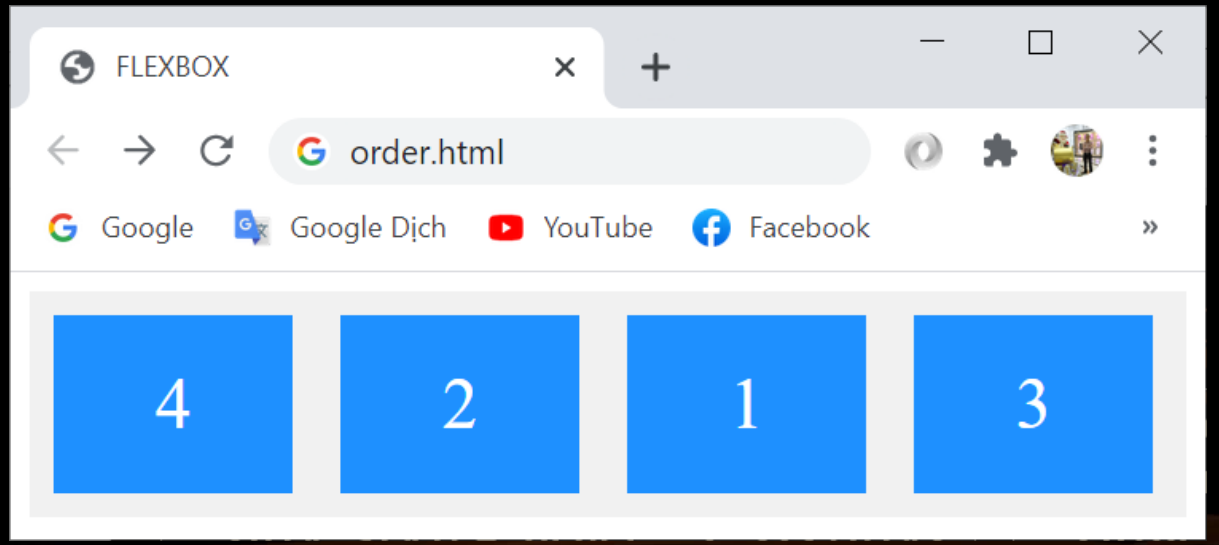
- **order** - order in which flex items appear in the flex container
  - <number> // *default 0*
- **flex-grow** - defines the ability of a flex item to grow if necessary
  - <number> // *default 0*
- **flex-shrink** - defines the ability of a flex item to shrink if necessary
  - <number> // *default 1*
- **flex-basis** - defines the default size of an element before the remaining space is distributed
  - <length> // *default auto*
- **flex** – shorthand for (**flex-grow**) & (**flex-shrink**) & (**flex-basis**)
- **align-self** – default alignment to be overridden for individual flex item

Flex: order

```
<style>
.flex-container {
  display: flex;
  align-items: stretch;
  background-color: #f1f1f1;
}
```

```
.flex-container>div {
  background-color: DodgerBlue;
  color: white;
  width: 100px;
  margin: 10px;
  text-align: center;
  line-height: 75px;
  font-size: 30px;
}
</style>
```

```
<div class="flex-container">
  <div style="order: 3">1</div>
  <div style="order: 2">2</div>
  <div style="order: 4">3</div>
  <div style="order: 1">4</div>
</div>
```

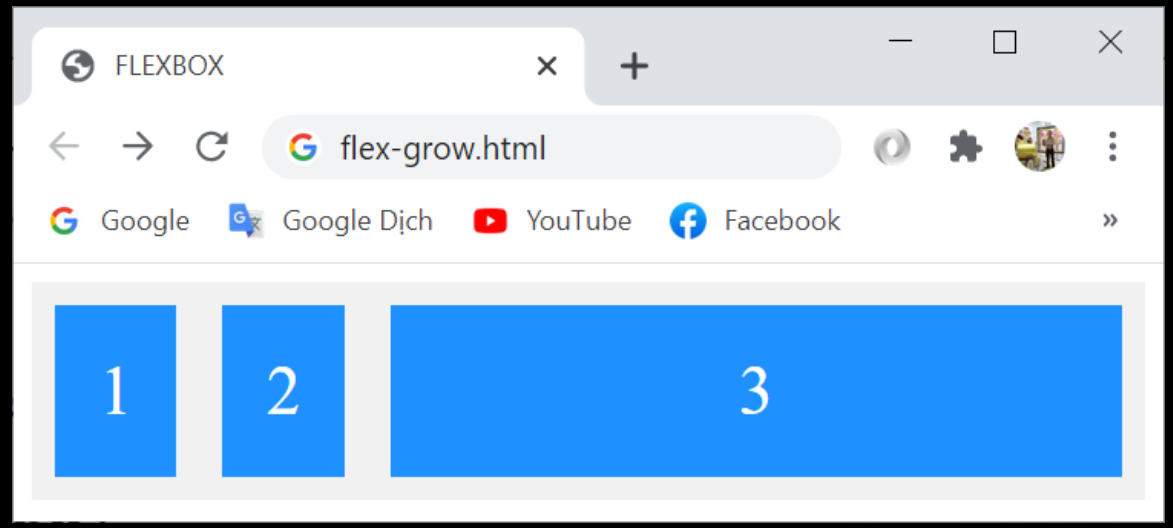


Flex: flex-grow

```
<style>
.flex-container {
  display: flex;
  align-items: stretch;
  background-color: #f1f1f1;
}
```

```
.flex-container > div {
  background-color: DodgerBlue;
  color: white;
  margin: 10px;
  text-align: center;
  line-height: 75px;
  font-size: 30px;
}
</style>
```

```
<div class="flex-container">
  <div style="flex-grow: 1">1</div>
  <div style="flex-grow: 1">2</div>
  <div style="flex-grow: 8">3</div>
</div>
```

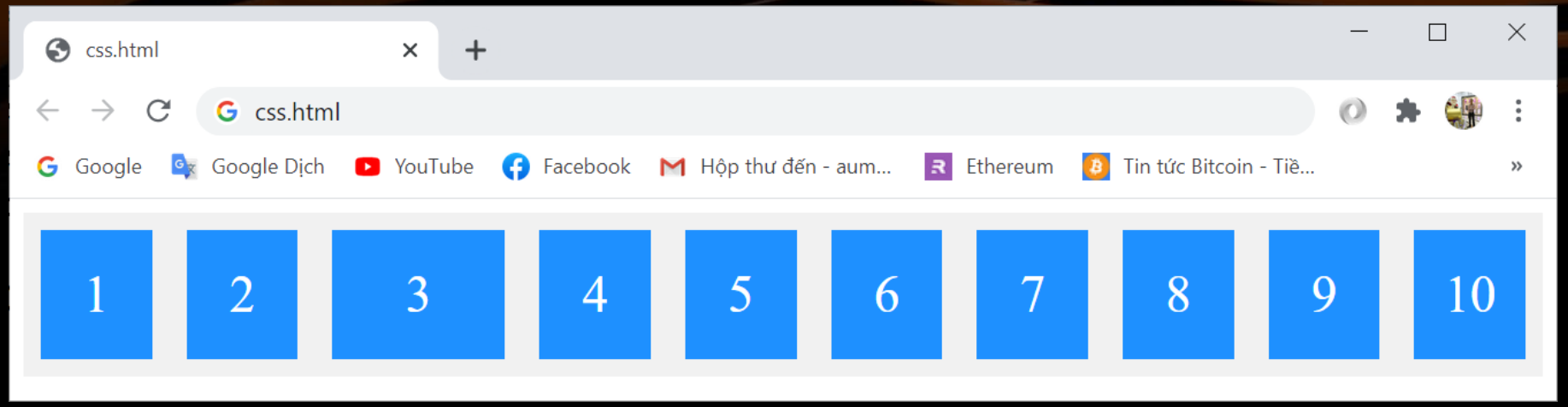


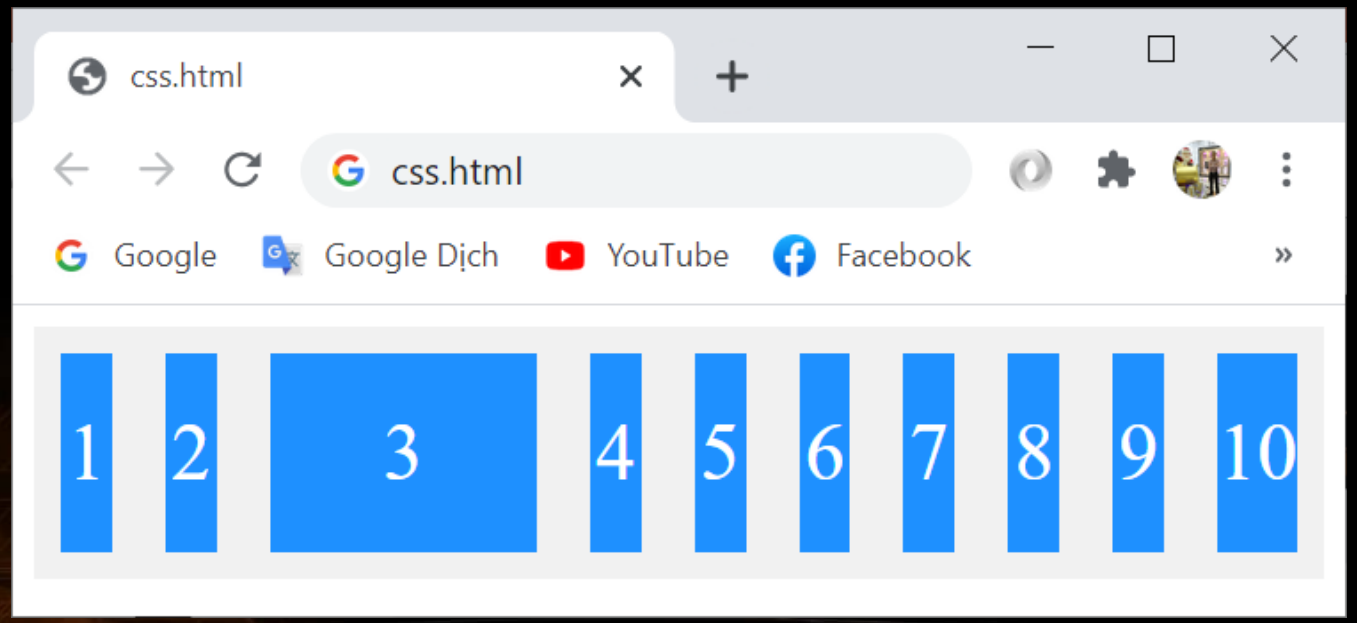
Flex: flex-shrink (1)

```
<div id="flex-container">
  <div class="flex-container">
  <div>1</div>
  <div>2</div>
  <div style="flex-shrink: 0">3</div>
  <div>4</div>
  <div>5</div>
  <div>6</div>
  <div>7</div>
  <div>8</div>
  <div>9</div>
  <div>10</div>
</div>
</div>
```

```
<style>
.flex-container {
  display: flex;
  align-items: stretch;
  background-color: #f1f1f1;
}
.flex-container>div {
  background-color: DodgerBlue;
  color: white;
  width: 100px;
  margin: 10px;
  text-align: center;
  line-height: 75px;
  font-size: 30px;
}
</style>
```

Flex: flex-shrink (2)



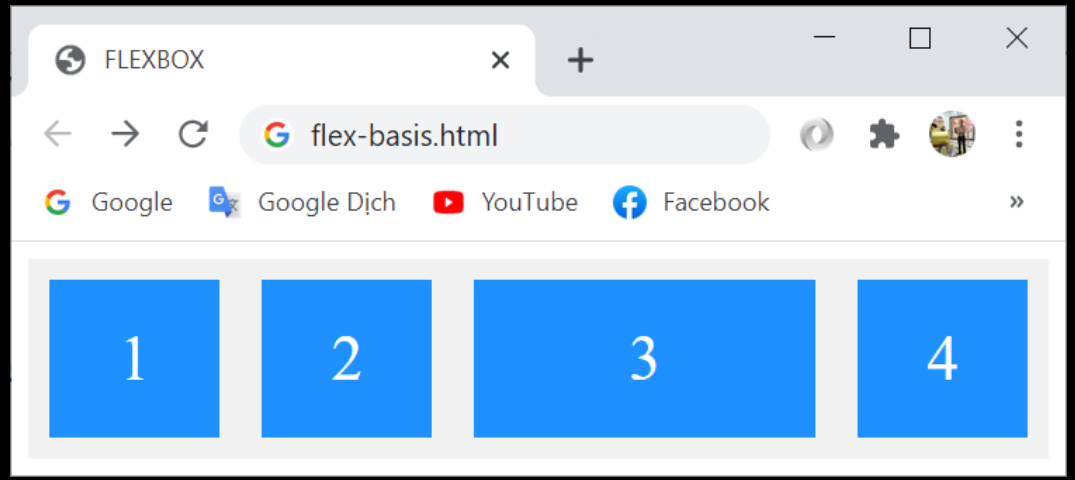


Flex: flex-basis

```
<style>
.flex-container {
  display: flex;
  align-items: stretch;
  background-color: #f1f1f1;
}
```

```
.flex-container > div {
  background-color: DodgerBlue;
  color: white;
  width: 100px;
  margin: 10px;
  text-align: center;
  line-height: 75px;
  font-size: 30px;
}
</style>
```

```
<div class="flex-container">
  <div>1</div>
  <div>2</div>
  <div style="flex-basis:200px">3</div>
  <div>4</div>
</div>
```



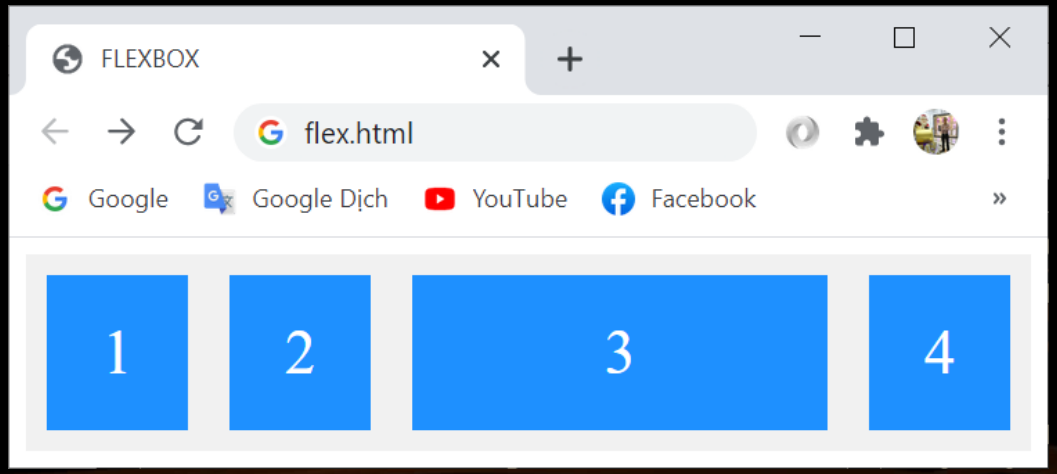
Flex: flex

flex – shorthand for (**flex-grow**) & (**flex-shrink**) & (**flex-basis**)

```
<style>
.flex-container {
  display: flex;
  align-items: stretch;
  background-color: #f1f1f1;
}
```

```
.flex-container>div {
  background-color: DodgerBlue;
  color: white;
  width: 100px;
  margin: 10px;
  text-align: center;
  line-height: 75px;
  font-size: 30px;
}
</style>
```

```
<div class="flex-container">
  <div>1</div>
  <div>2</div>
  <div style="flex: 0 0 200px">3</div>
  <div>4</div>
</div>
```

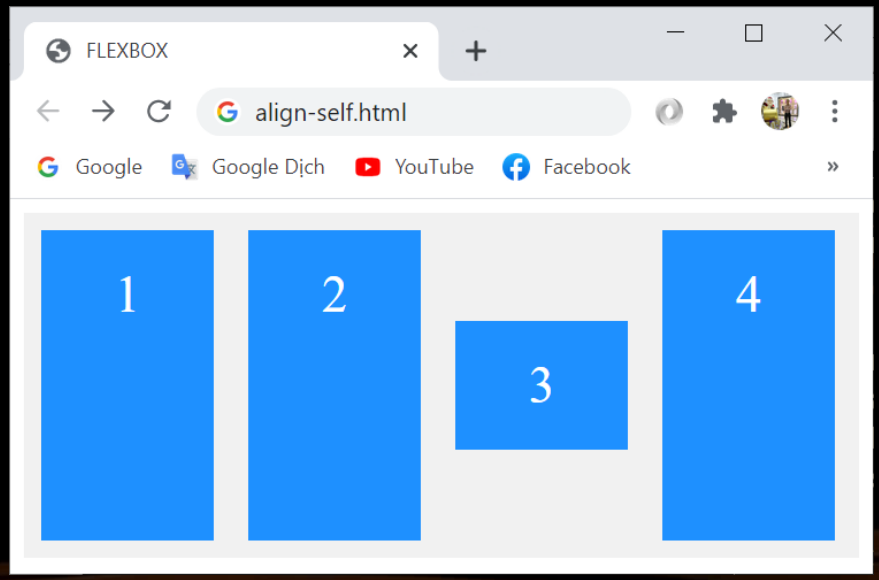


Flex: align-self

```
<style>
.flex-container {
  display: flex;
  height: 200px;
  background-color: #f1f1f1;
}
```

```
.flex-container > div {
  background-color: DodgerBlue;
  color: white;
  width: 100px;
  margin: 10px;
  text-align: center;
  line-height: 75px;
  font-size: 30px;
}
</style>
```

```
<div class="flex-container">
  <div>1</div>
  <div>2</div>
  <div style="align-self: center">3</div>
  <div>4</div>
</div>
```



Grid Layout

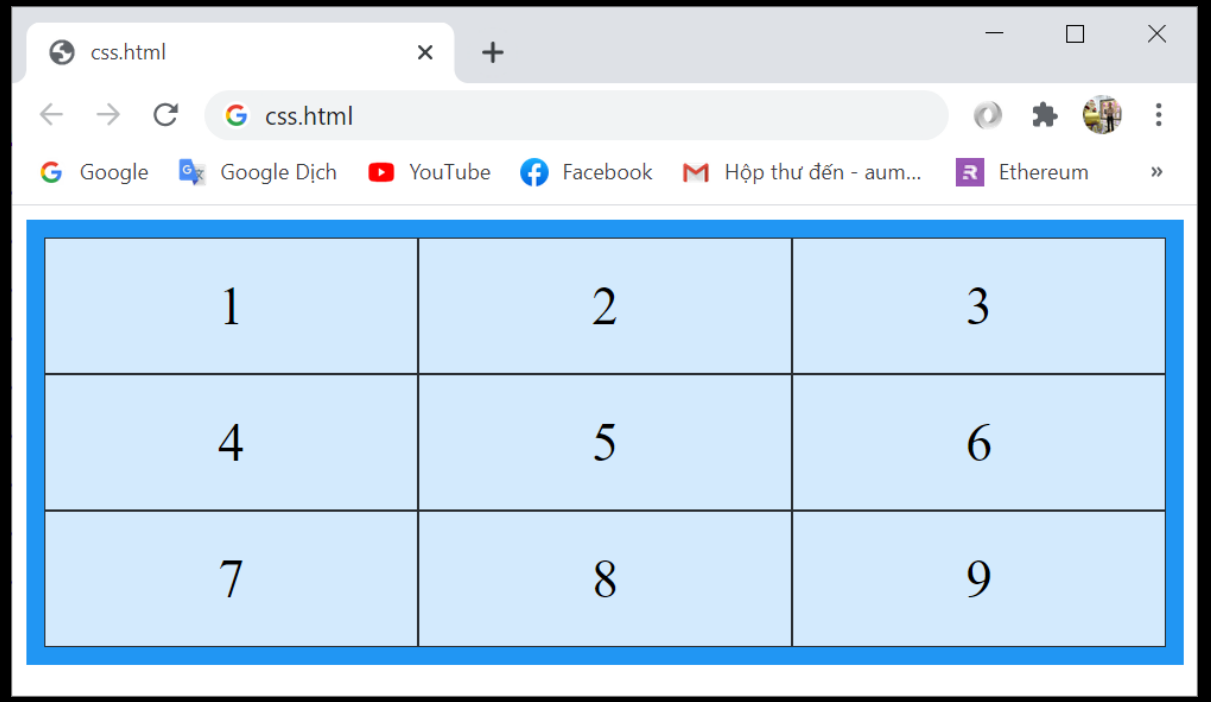
- The CSS Grid Layout Module offers a grid-based layout system, with rows and columns, making it easier to design web pages without having to use floats and positioning.
- A grid layout consists of a parent element, with one or more child elements.

Grid Elements – Example (1)

```
<style>
.grid-container {
  display: grid;
  grid-template-columns: auto auto auto;
  background-color: #2196F3;
  padding: 10px;
}
.grid-item {
  background-color: rgba(255, 255, 255, 0.8);
  border: 1px solid rgba(0, 0, 0, 0.8);
  padding: 20px;
  font-size: 30px;
  text-align: center;
}
</style>
```

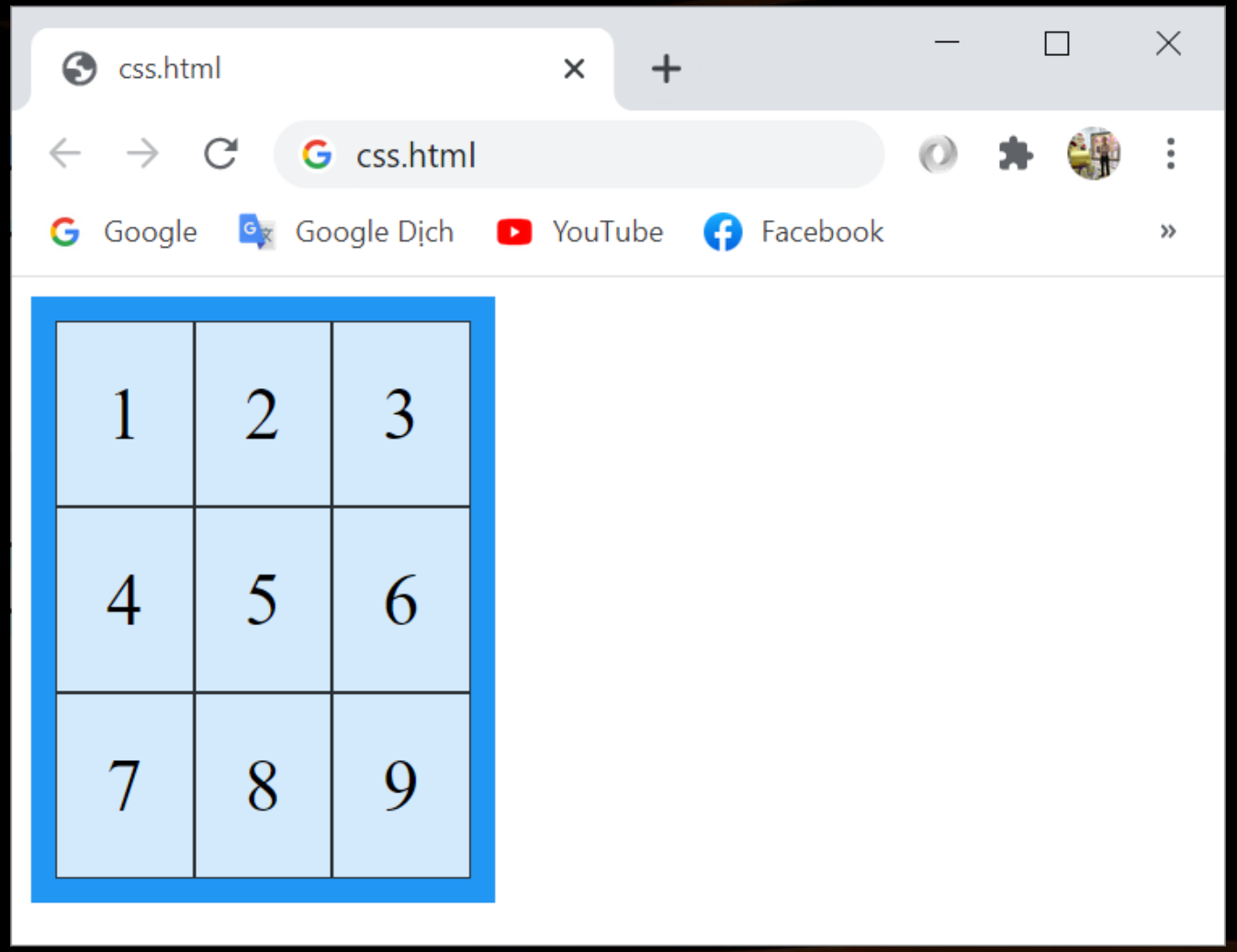
Grid Elements – Example (2)

```
<div class="grid-container">
  <div class="grid-item">1</div>
  <div class="grid-item">2</div>
  <div class="grid-item">3</div>
  <div class="grid-item">4</div>
  <div class="grid-item">5</div>
  <div class="grid-item">6</div>
  <div class="grid-item">7</div>
  <div class="grid-item">8</div>
  <div class="grid-item">9</div>
</div>
```

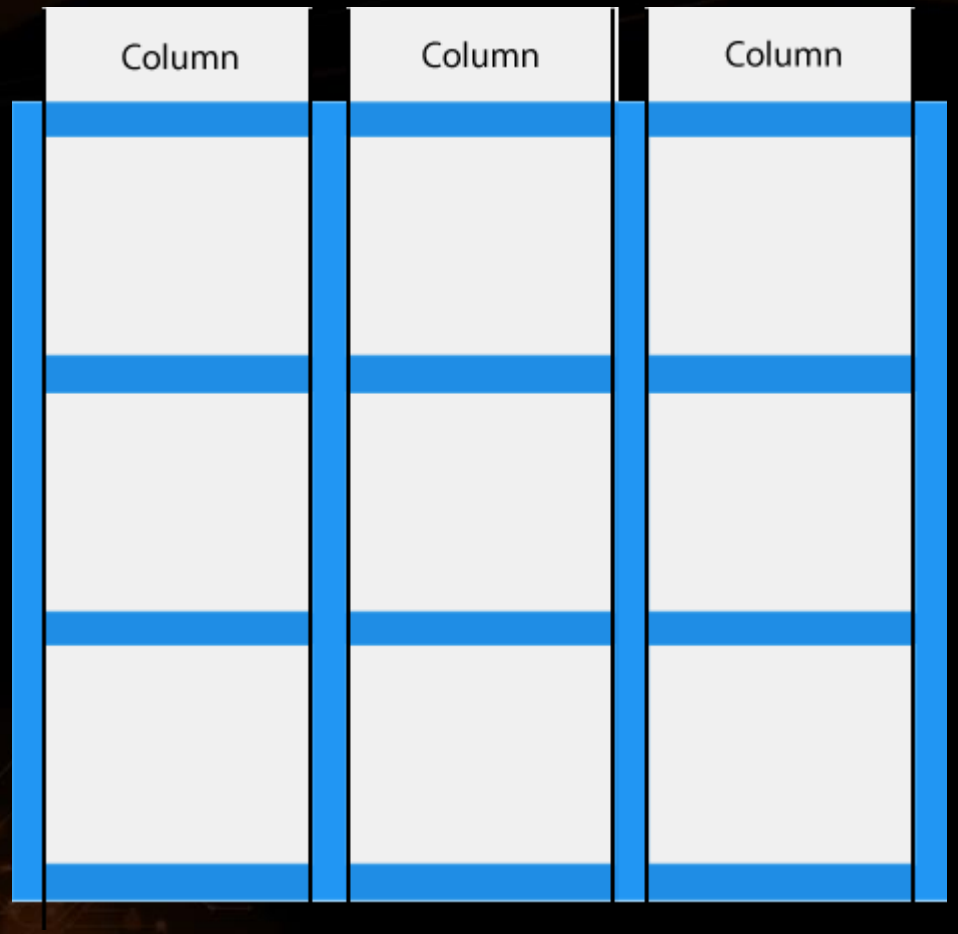


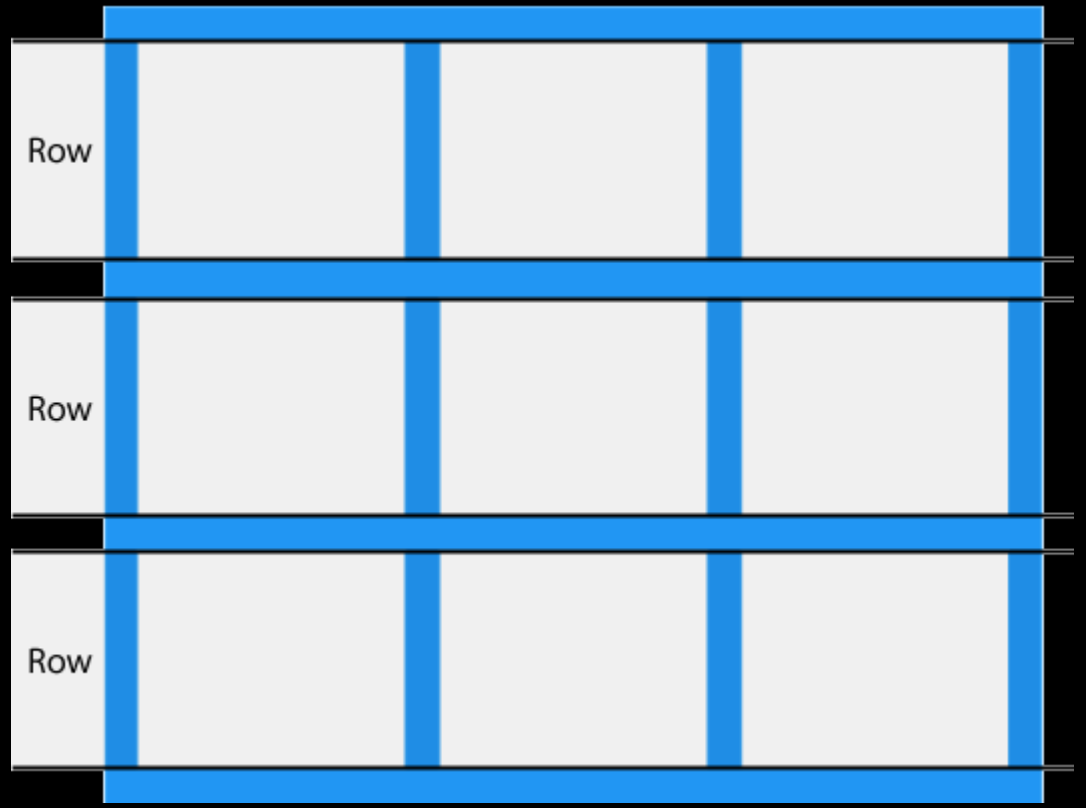
Display Property

```
<style>
.grid-container {
  display: inline-grid;
}
</style>
```



Grid Columns and Rows





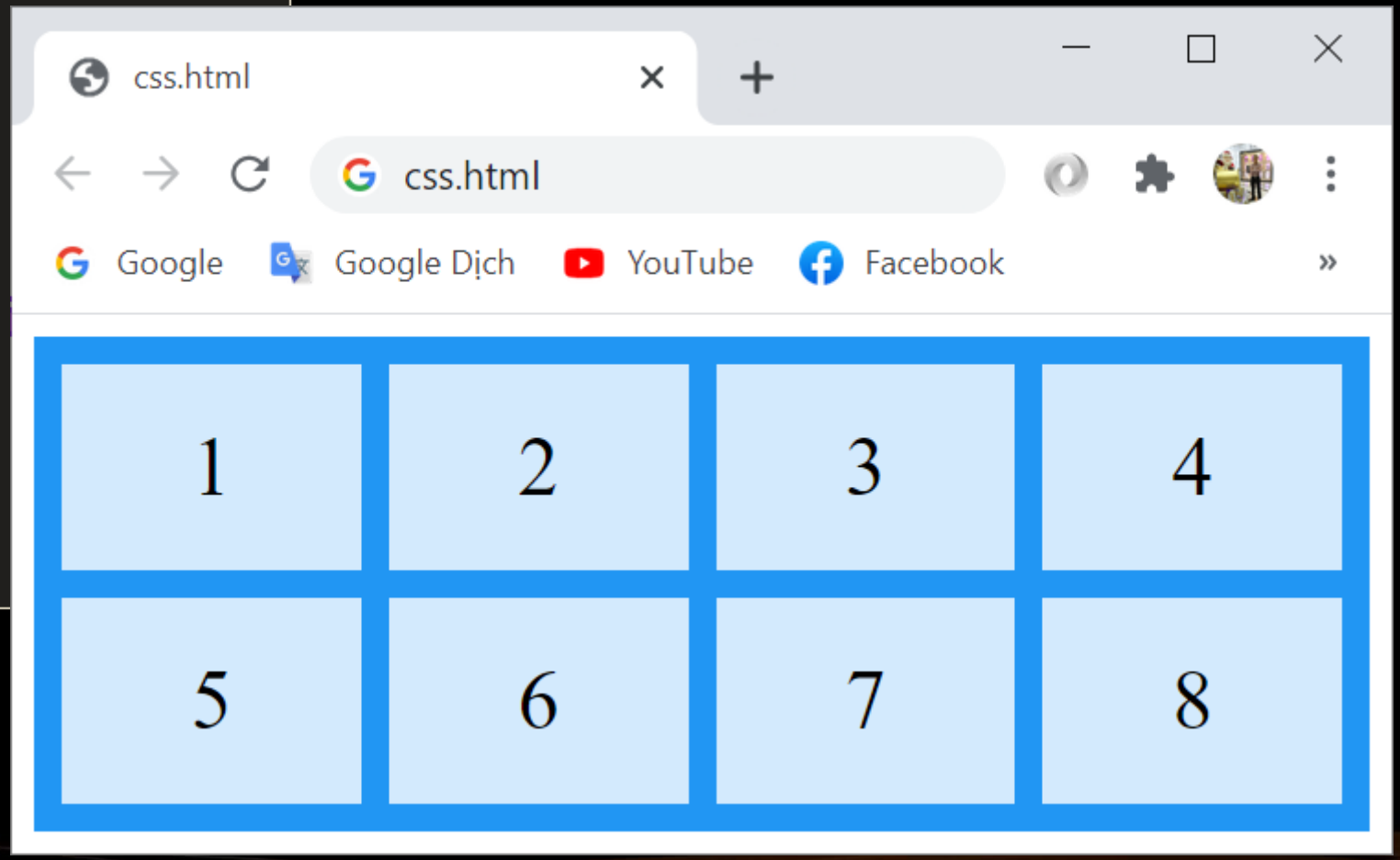
grid-template-columns – Example (1)

- The grid-template-columns property defines the number of columns in your grid layout, and it can define the width of each column.

```
<style>
.grid-container {
  display: grid;
  grid-template-columns: auto auto auto auto;
  grid-gap: 10px;
  background-color: #2196F3;
  padding: 10px;
}
.grid-container > div {
  background-color: rgba(255, 255, 255, 0.8);
  text-align: center;
  padding: 20px 0;
  font-size: 30px;
}
</style>
```

grid-template-columns – Example (2)

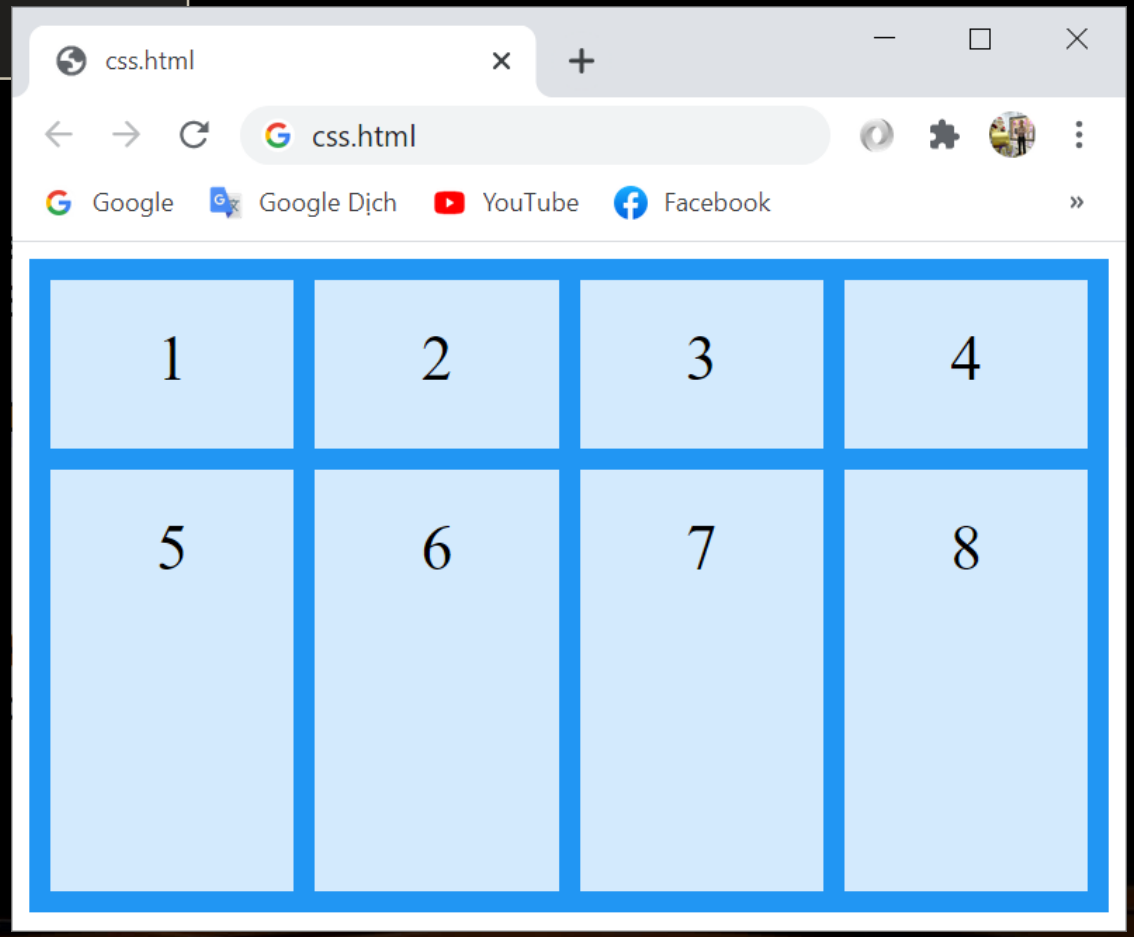
```
<div class="grid-container">
  <div>1</div>
  <div>2</div>
  <div>3</div>
  <div>4</div>
  <div>5</div>
  <div>6</div>
  <div>7</div>
  <div>8</div>
</div>
```



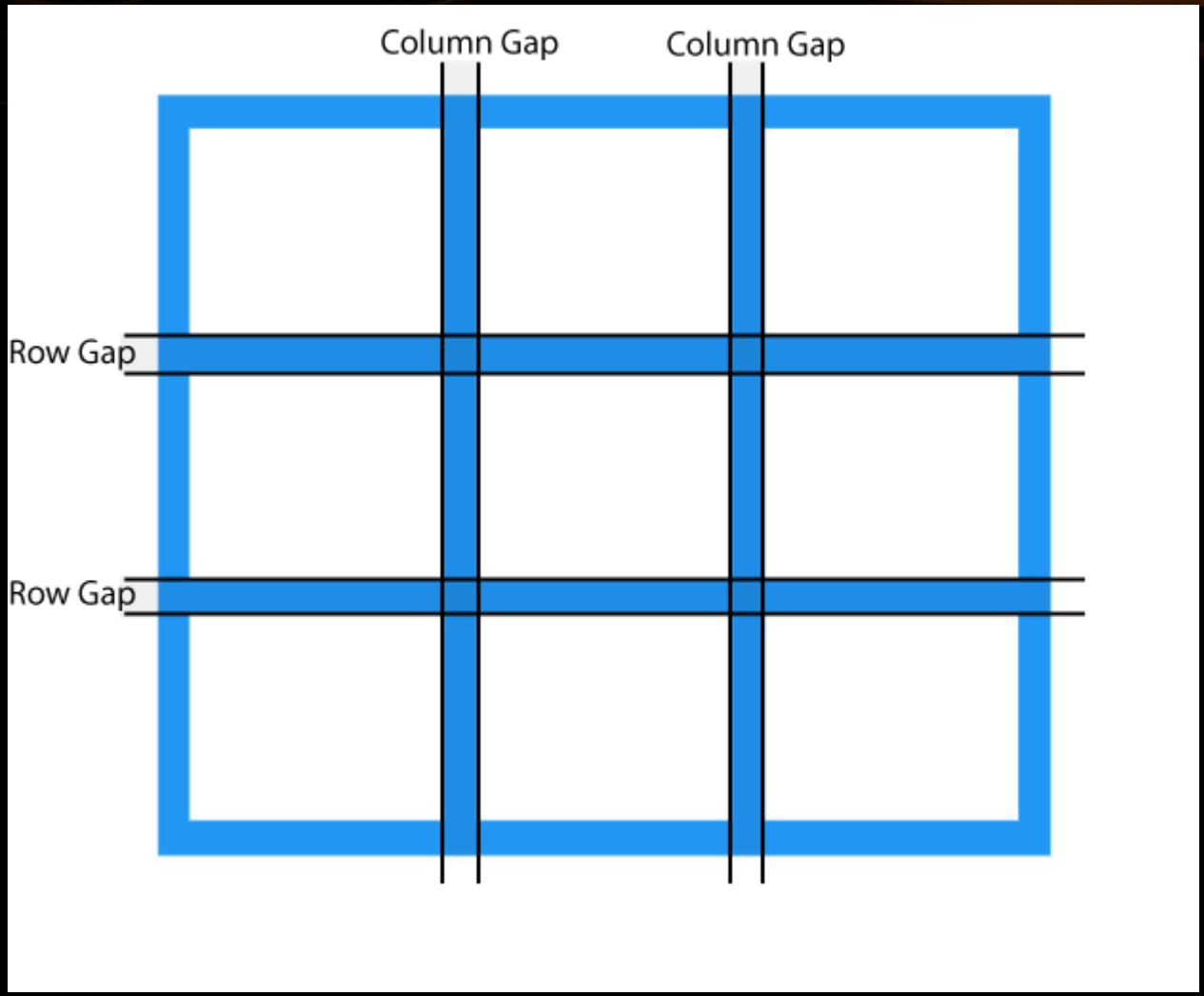
grid-template-rows – Example

```
.grid-container {
    grid-template-rows: 80px 200px;
}
```

- The grid-template-rows property defines the height of each row.

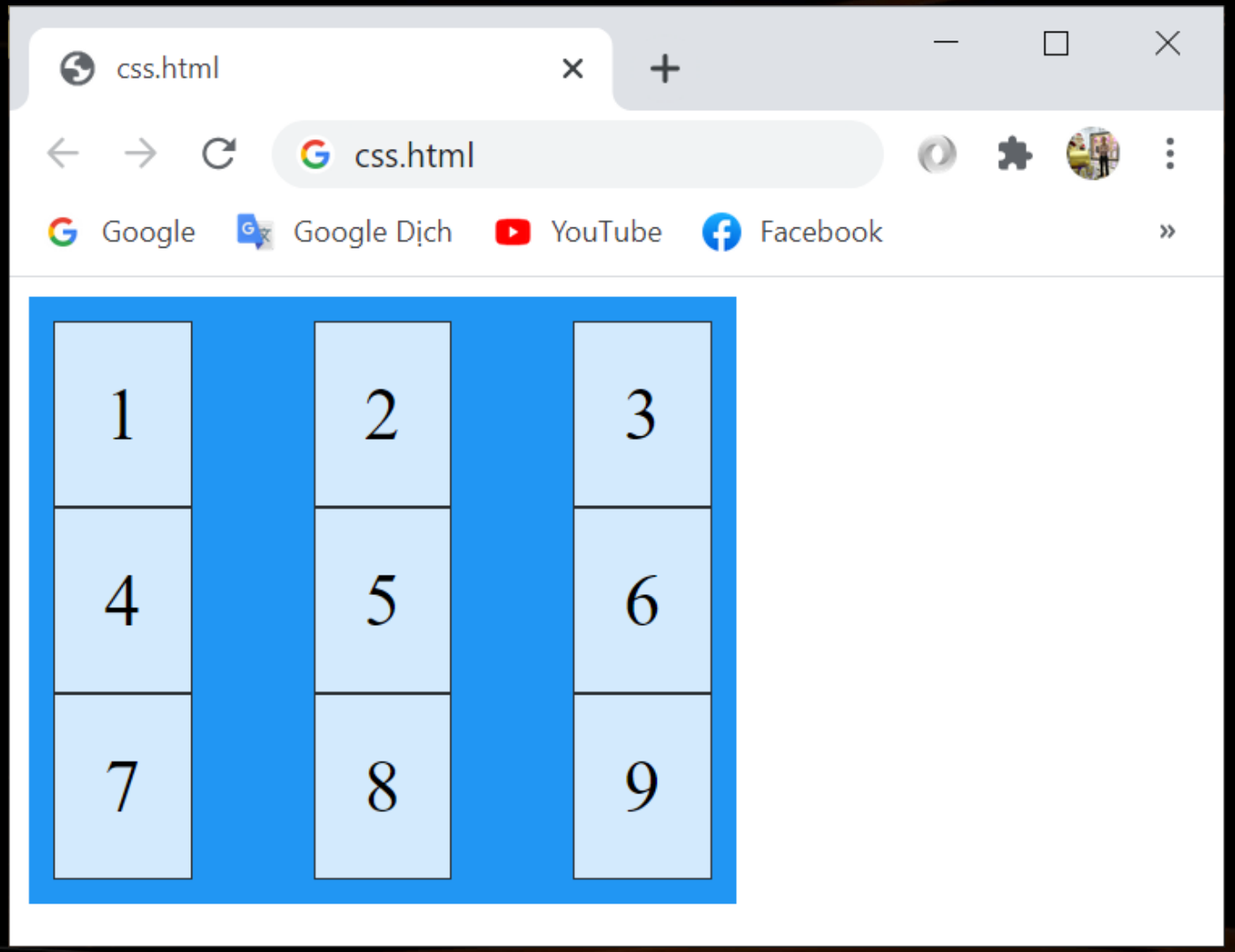


Grid Gaps



Grid Column Gap – Example

```
<style>
.grid-container {
  grid-column-gap: 50px;
}
</style>
```

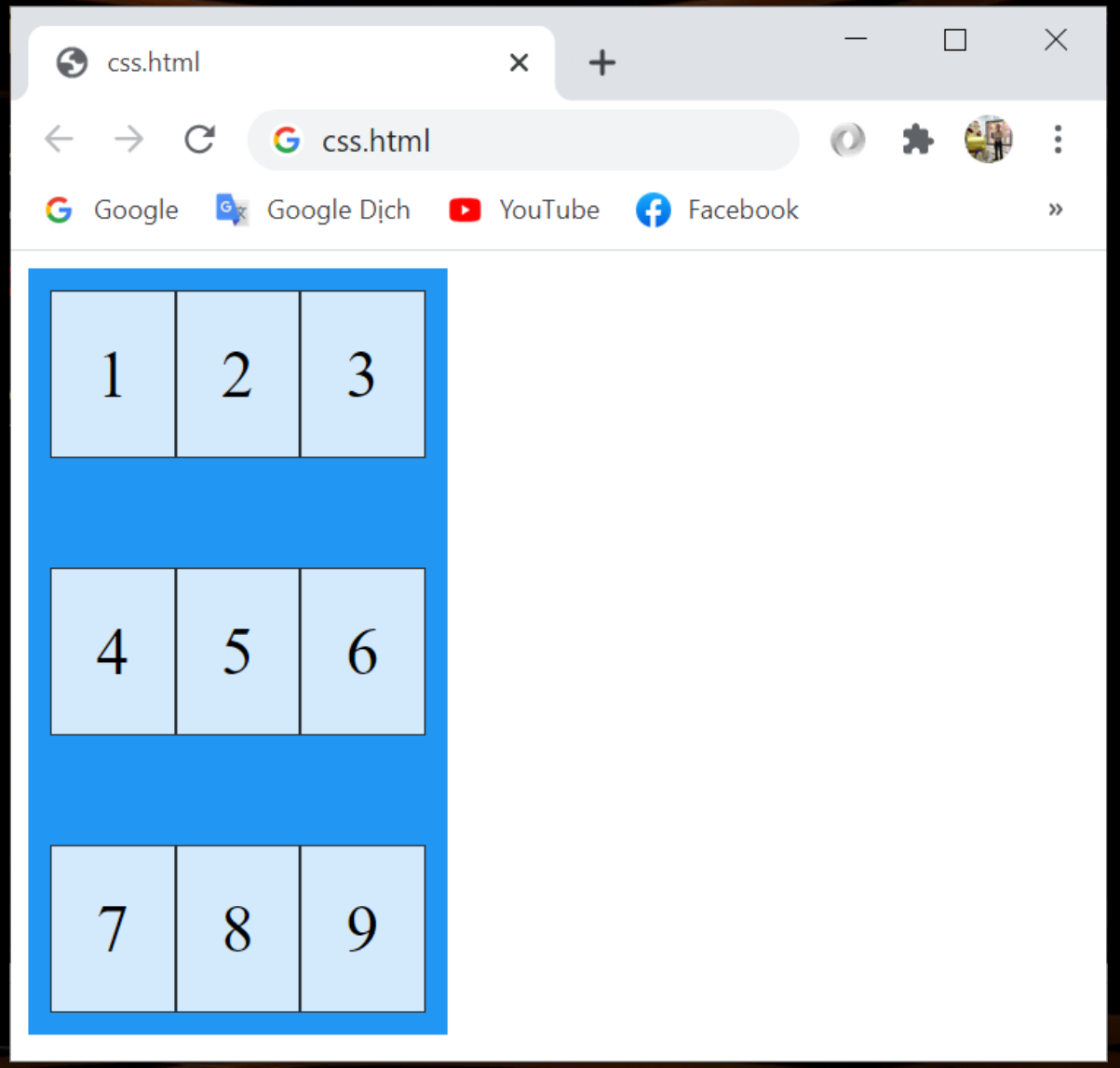


- The grid-column-gap property sets the gap between the columns

Grid Row Gap – Example

```
<style>
.grid-container {
  grid-row-gap: 50px;
}
</style>
```

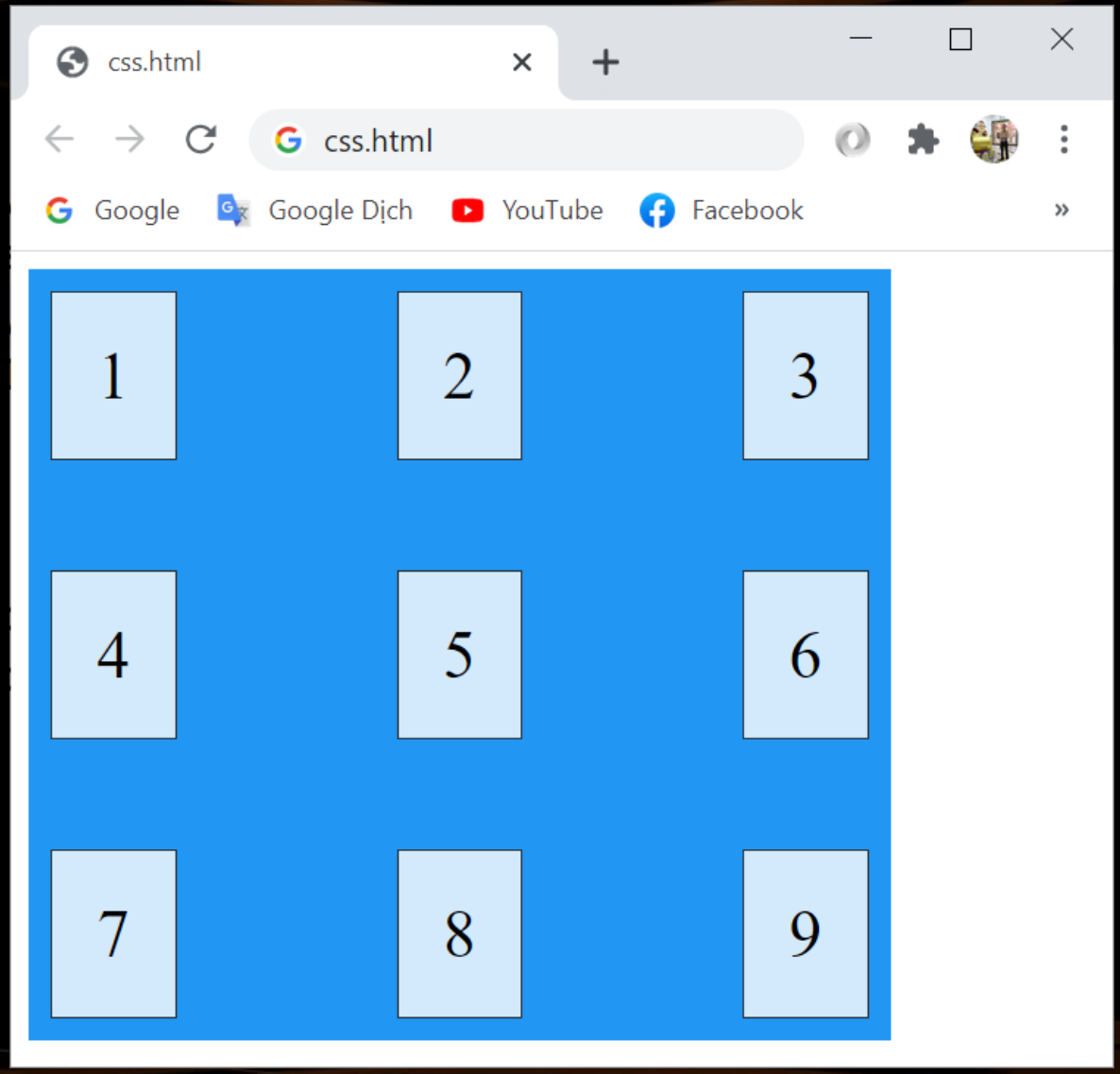
- The grid-row-gap property sets the gap between the rows



Grid Gaps - Shorthand Example

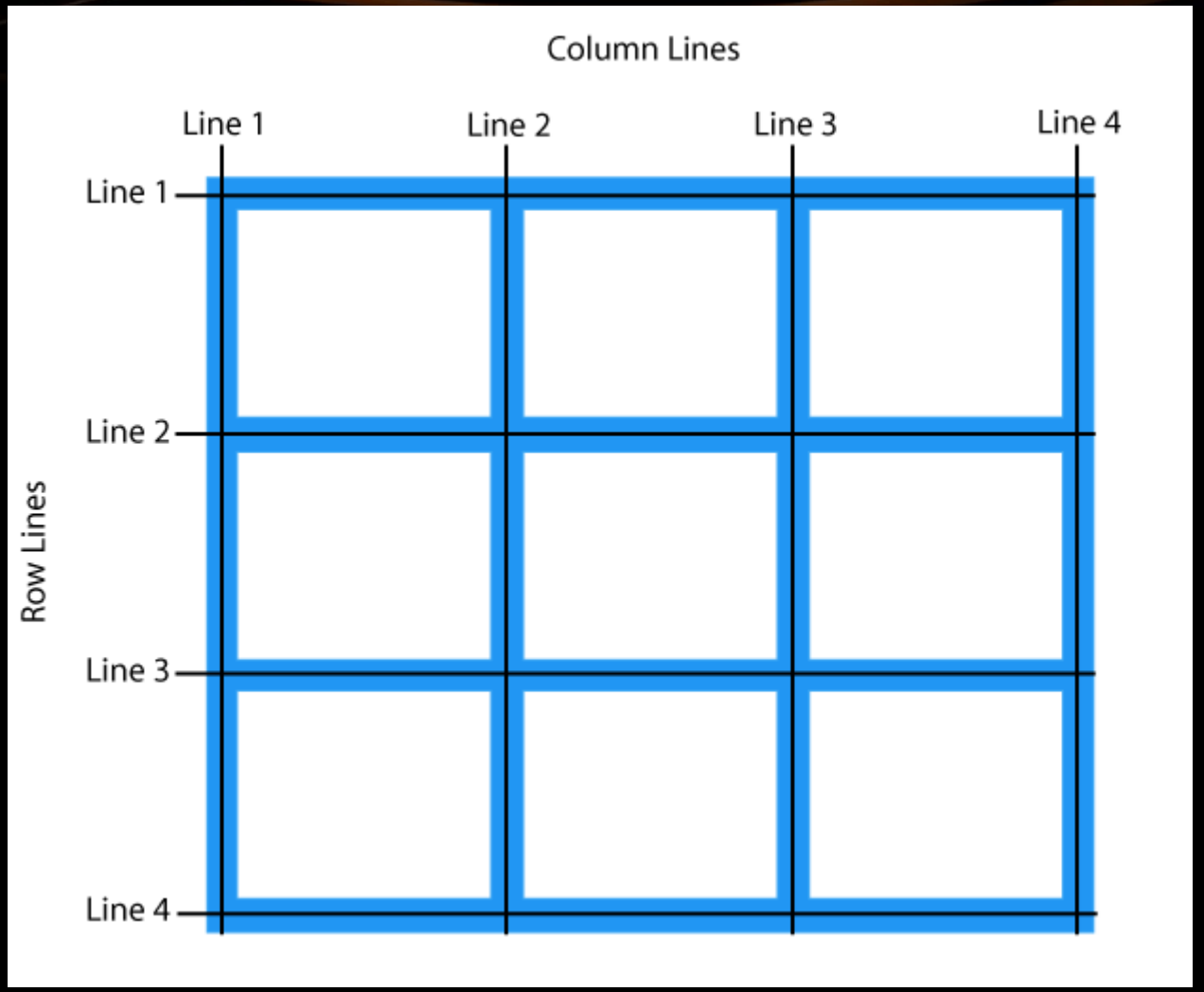
```
<style>
.grid-container {
  grid-gap: 50px 100px;
}
</style>
```

- grid-gap: grid-row-gap & grid-column-gap



Grid Lines

- The lines between columns are called column lines.
- The lines between rows are called row lines.



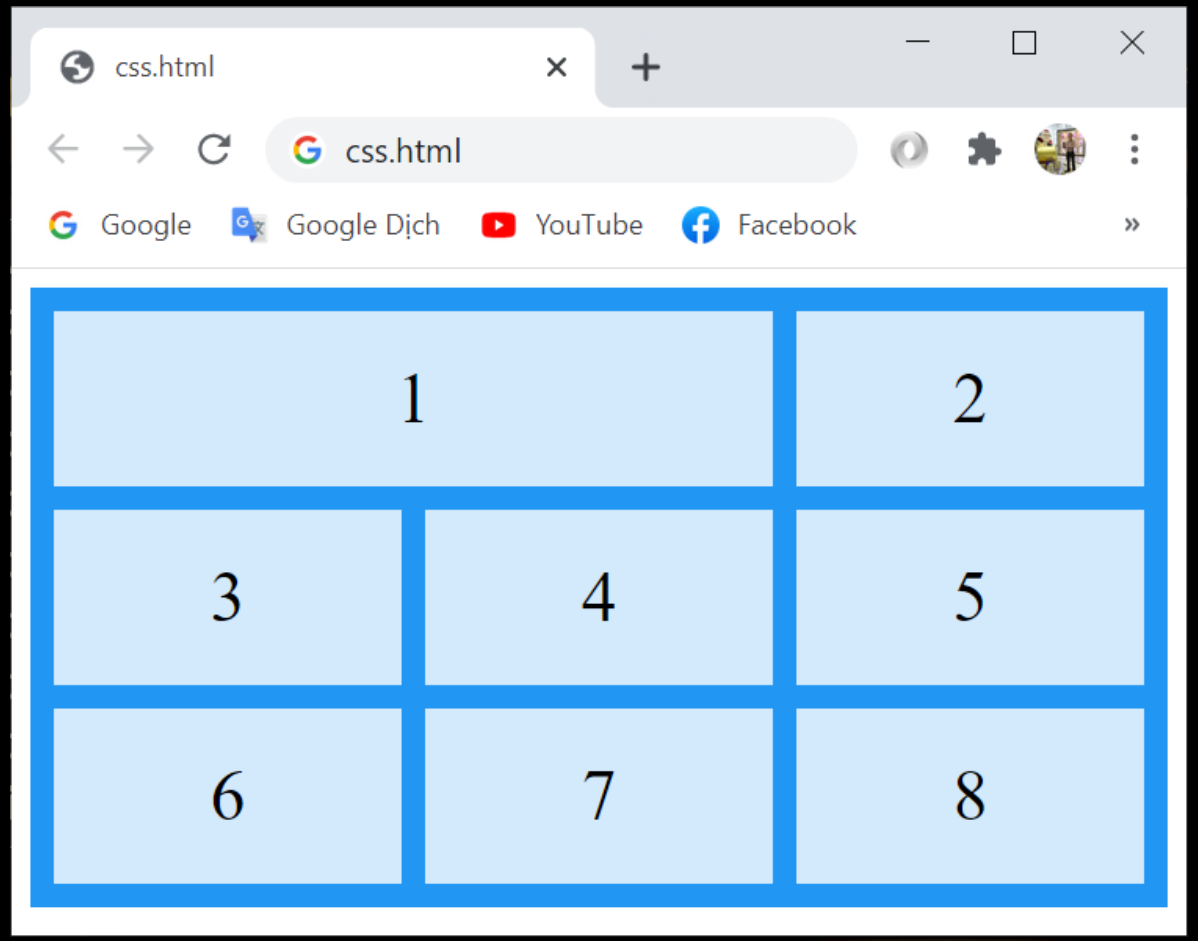
Grid Lines – column – Example (1)

```
<style>
.grid-container {
  display: grid;
  grid-template-columns: auto auto auto;
  grid-gap: 10px;
  background-color: #2196F3;
  padding: 10px;
}
.grid-container > div {
  background-color: rgba(255, 255, 255,
0.8);
  text-align: center;
  padding: 20px 0;
  font-size: 30px;
}
```

```
.item1 {
  grid-column-start: 1;
  grid-column-end: 3;
}
</style>
```

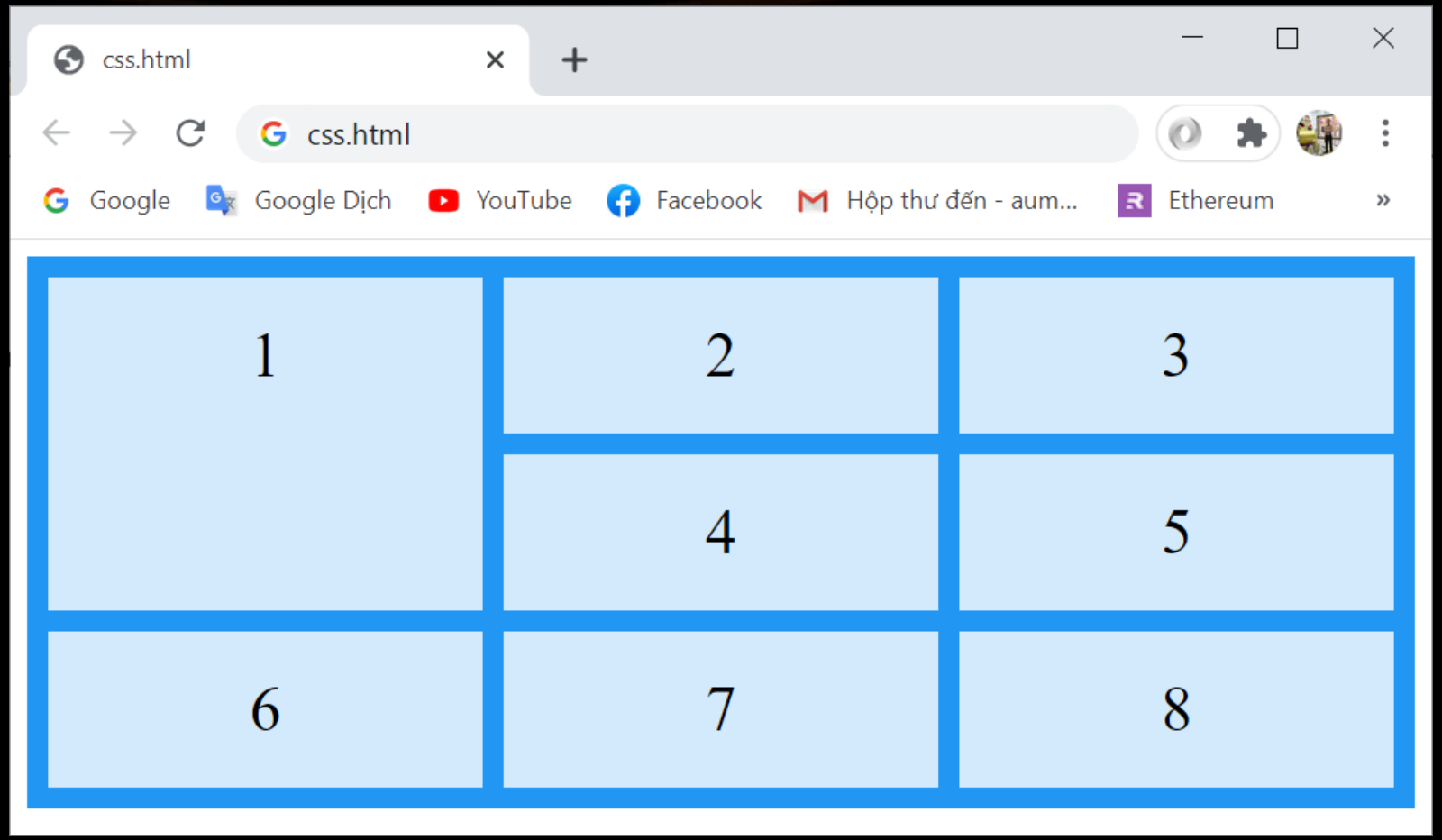
Grid Lines – column – Example (2)

```
<div class="grid-container">
  <div class="item1">1</div>
  <div class="item2">2</div>
  <div class="item3">3</div>
  <div class="item4">4</div>
  <div class="item5">5</div>
  <div class="item6">6</div>
  <div class="item7">7</div>
  <div class="item8">8</div>
</div>
```



Grid Lines – row – Example

```
.item1 {
  grid-row-start: 1;
  grid-row-end: 3;
}
```



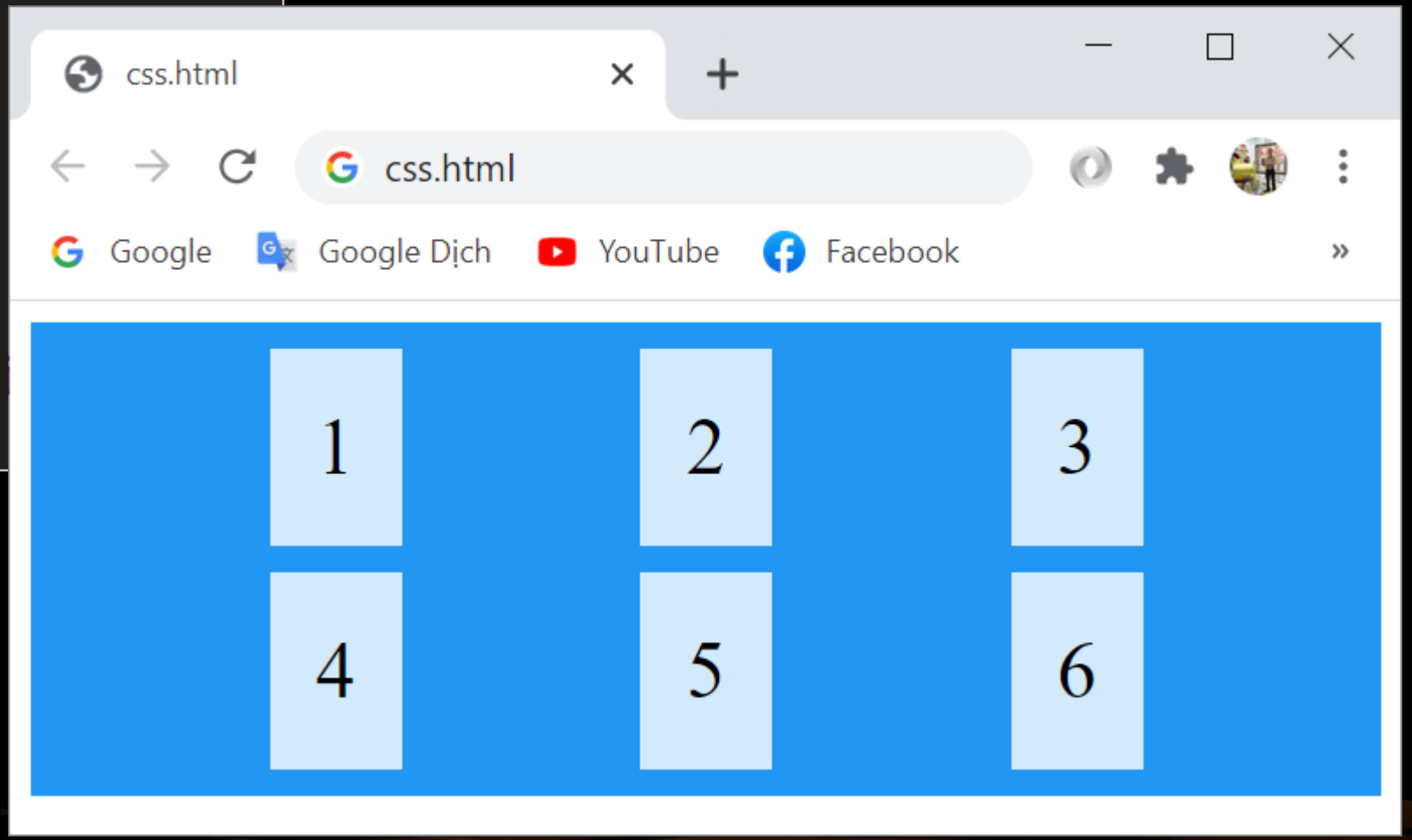
Grid – justify-content – space-evenly (1)

- The justify-content property is used to align the whole grid inside the container.

```
<style>
.grid-container {
  display: grid;
  justify-content: space-evenly;
  grid-template-columns: 50px 50px 50px;
  grid-gap: 10px;
  background-color: #2196F3;
  padding: 10px;
}
.grid-container > div {
  background-color: rgba(255, 255, 255, 0.8);
  text-align: center;
  padding: 20px 0;
  font-size: 30px;
}
</style>
```

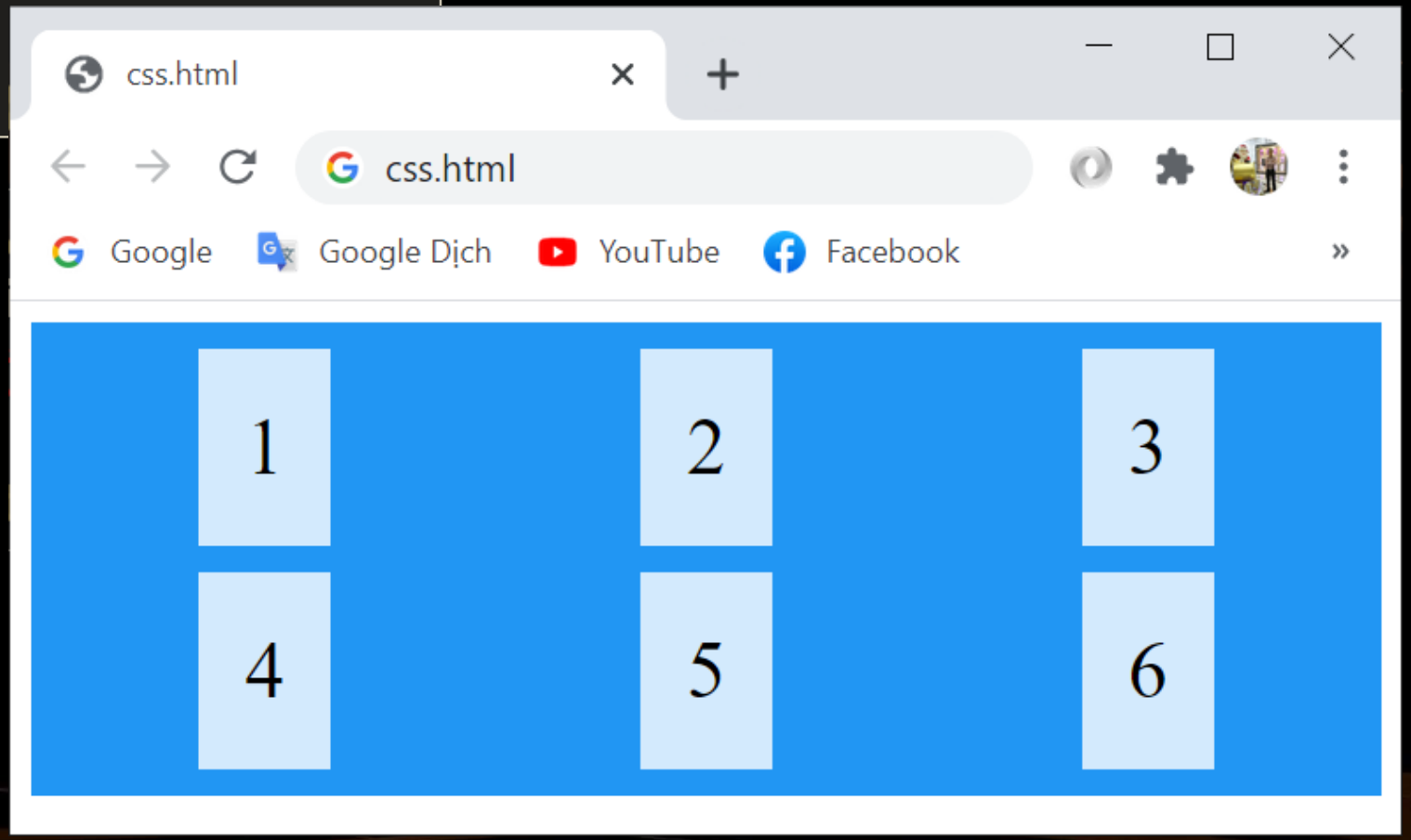
Grid – justify-content – space-evenly (2)

```
<div class="grid-container">
  <div>1</div>
  <div>2</div>
  <div>3</div>
  <div>4</div>
  <div>5</div>
  <div>6</div>
</div>
```



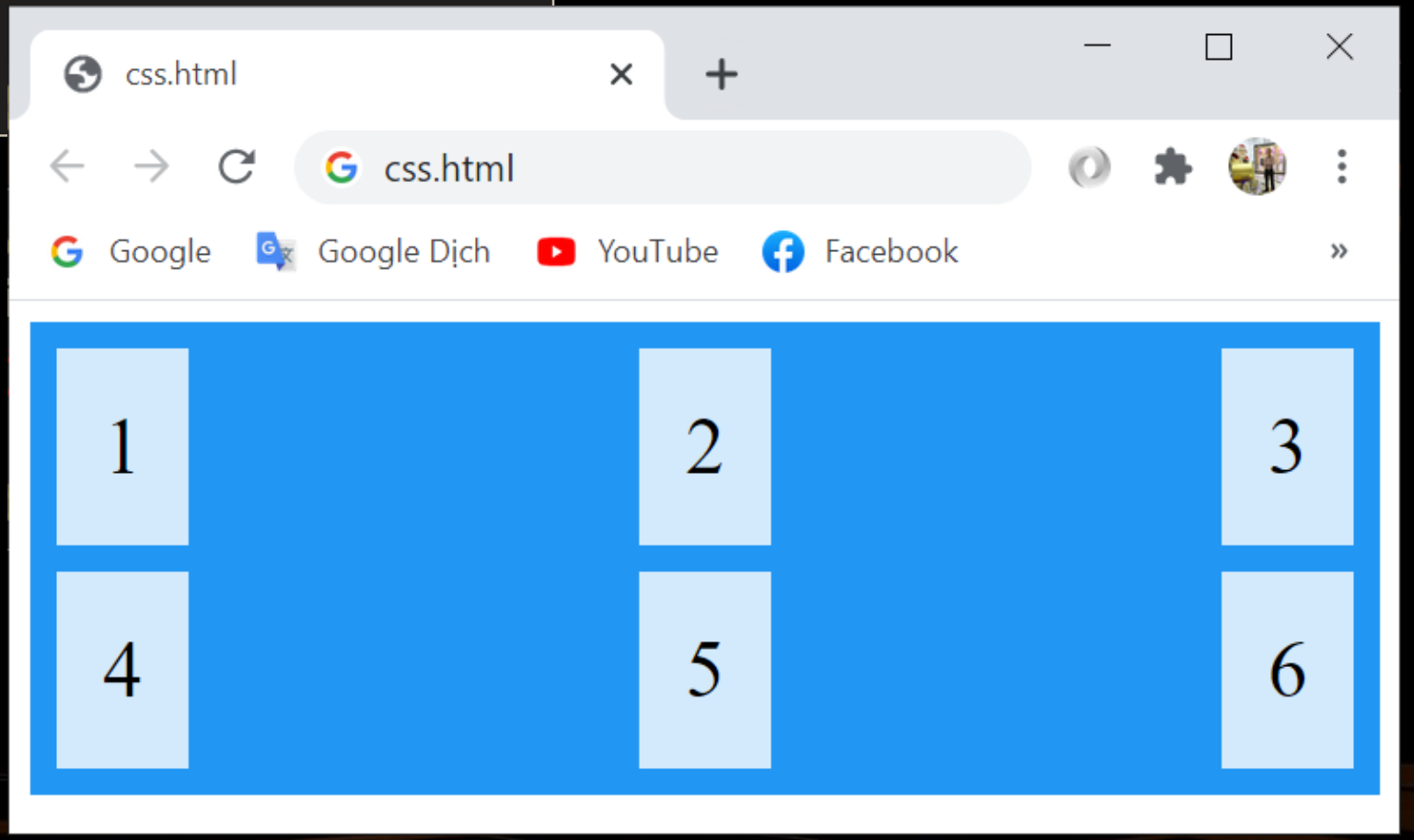
Grid – justify-content – space-around

```
<style>
.grid-container {
    justify-content: space-around;
}
</style>
```



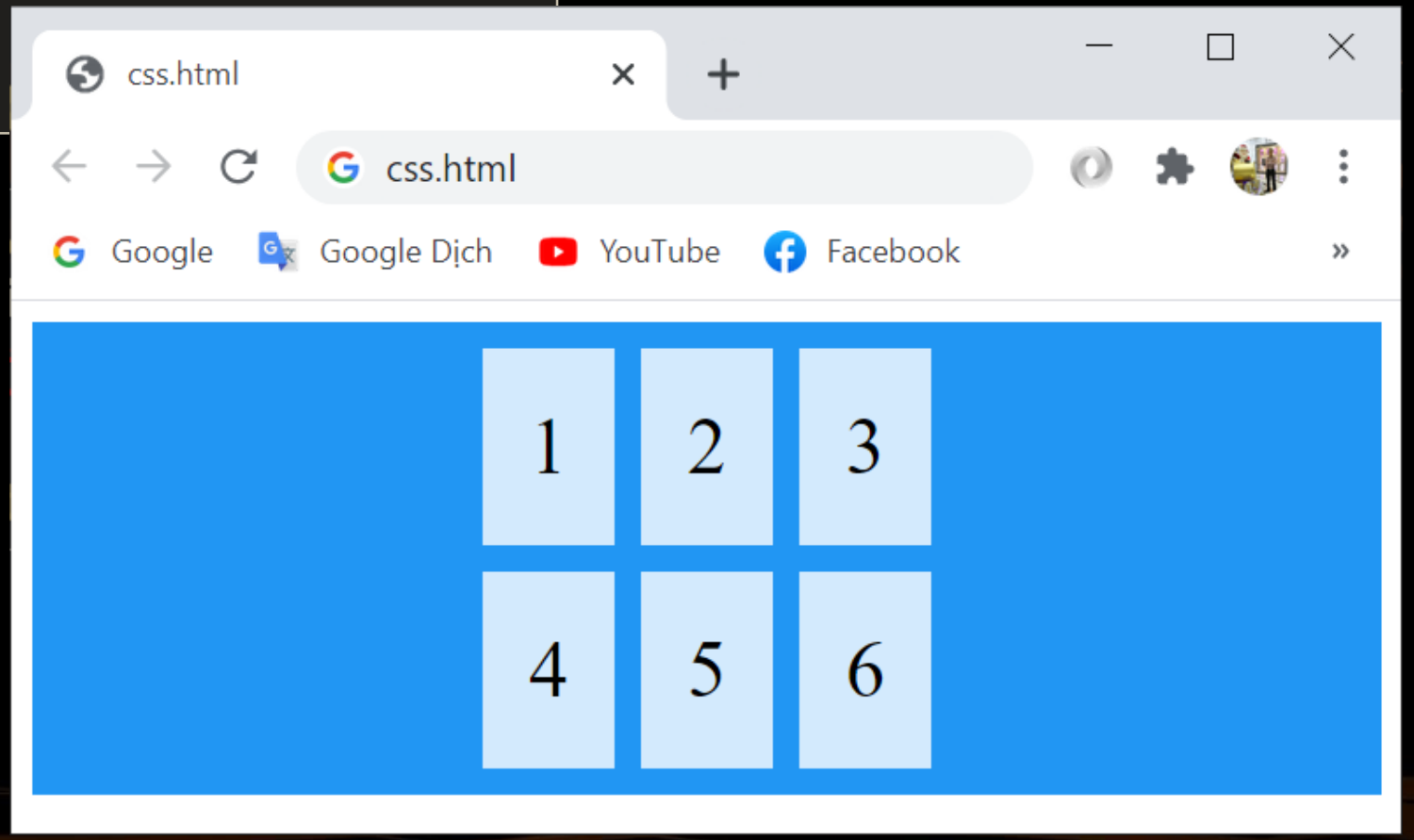
Grid – justify-content – space-between

```
<style>
.grid-container {
    justify-content: space-between;
}
</style>
```



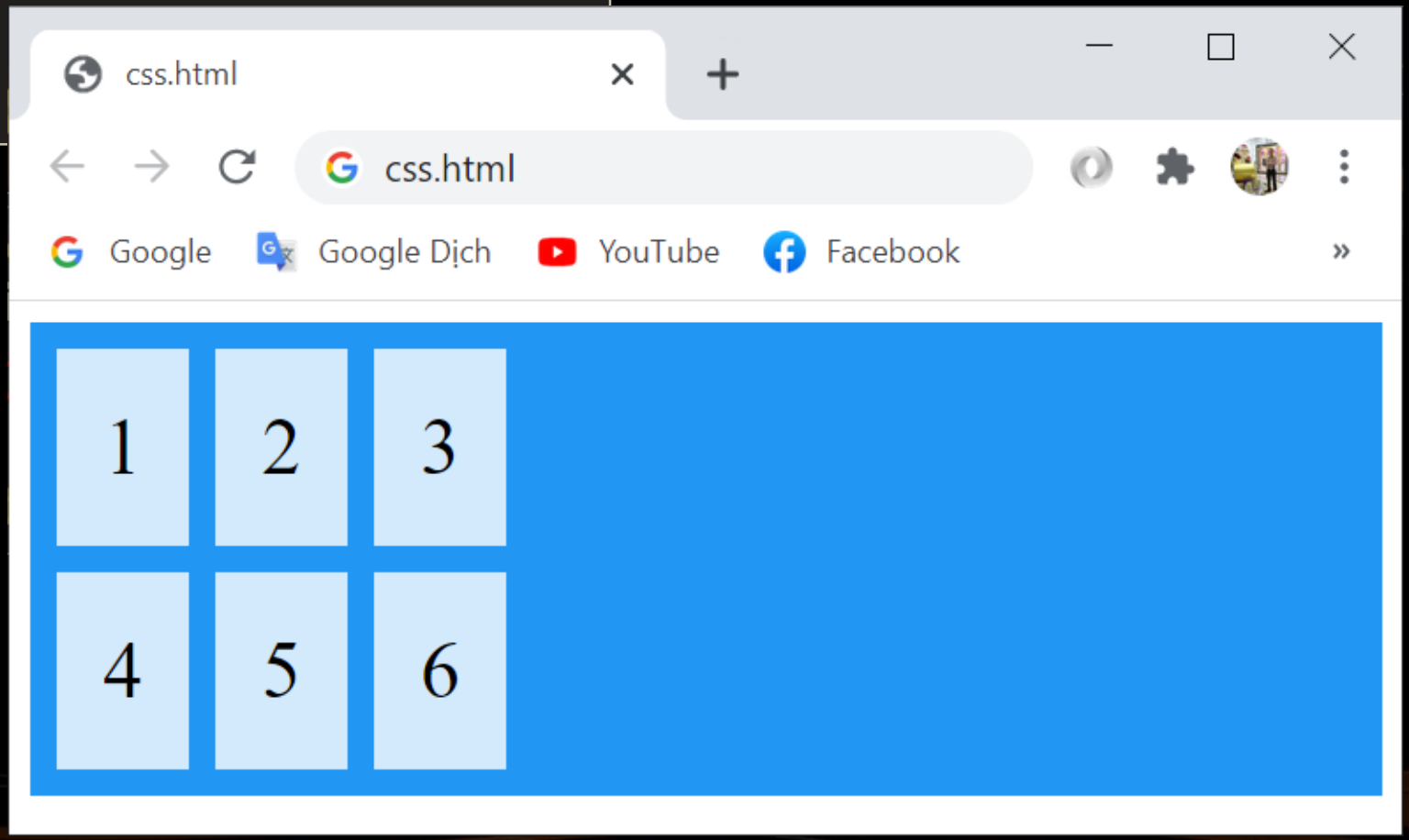
Grid – justify-content – center

```
<style>
.grid-container {
    justify-content: center;
}
</style>
```



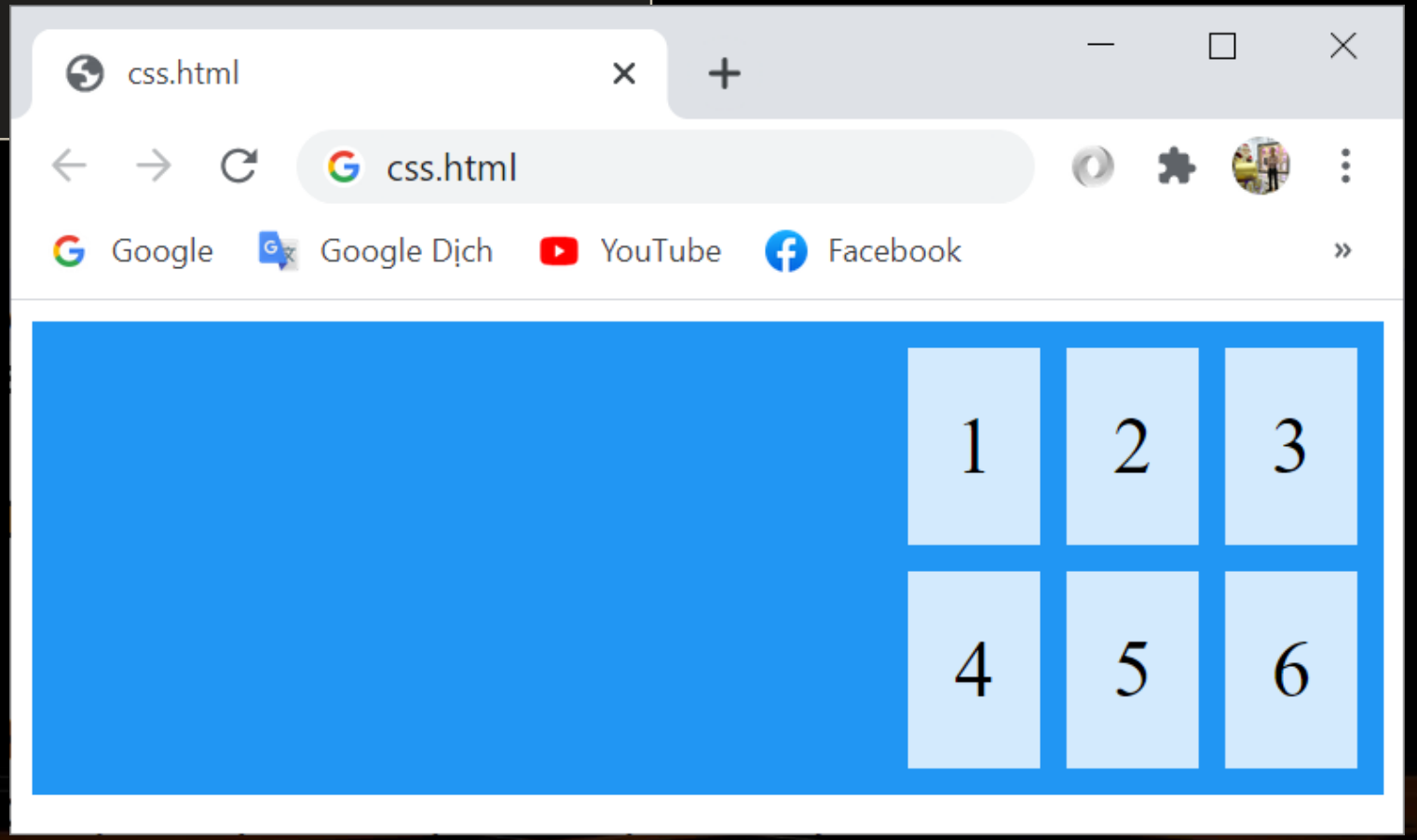
Grid – justify-content – start

```
<style>
.grid-container {
    justify-content: start;
}
</style>
```



Grid – justify-content – end

```
<style>
.grid-container {
    justify-content: end;
}
</style>
```



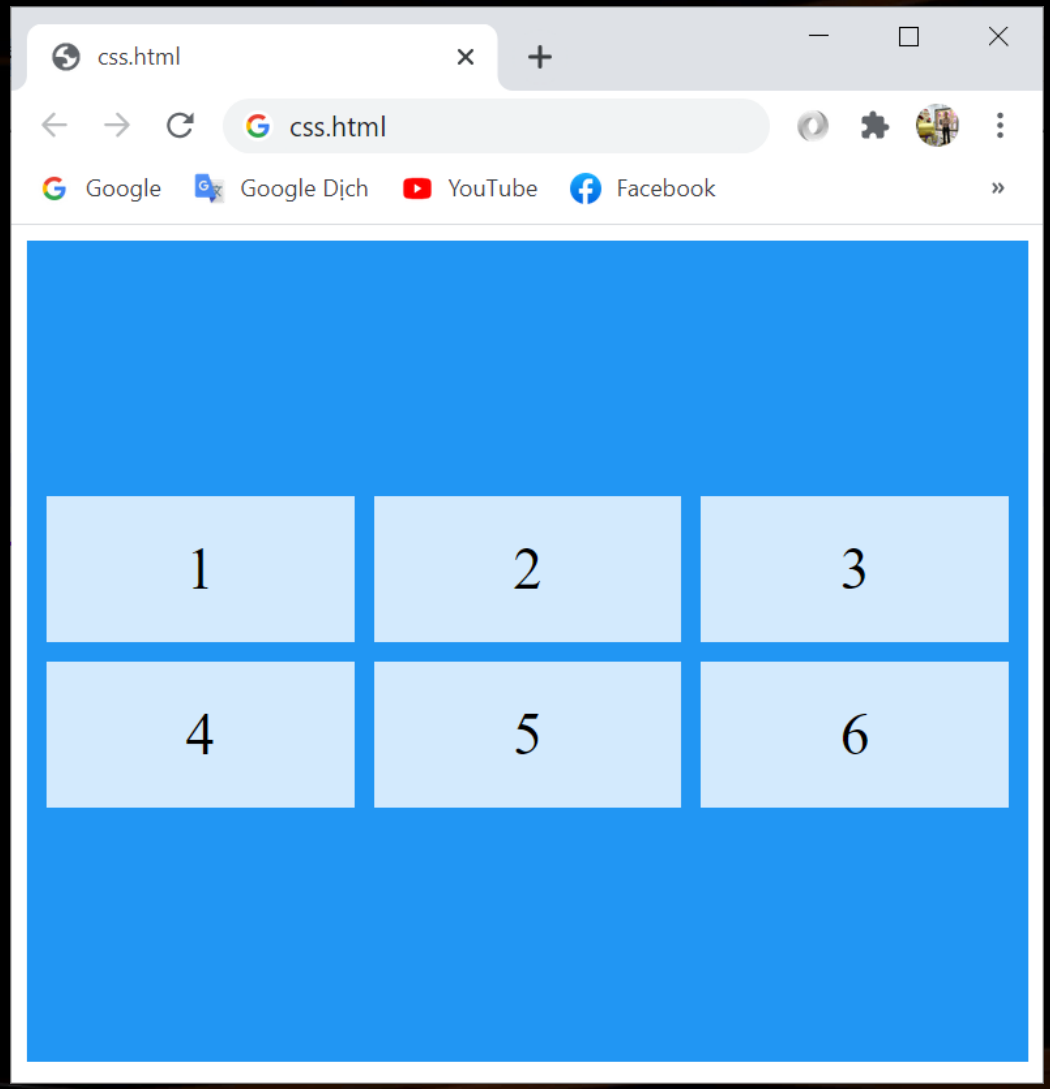
Grid – align-content – center (1)

- The align-content property is used to vertically align the whole grid inside the container

```
<style>
.grid-container {
  display: grid;
  height: 400px;
  align-content: center;
  grid-template-columns: auto auto auto;
  grid-gap: 10px;
  background-color: #2196F3;
  padding: 10px;
}
.grid-container > div {
  background-color: rgba(255, 255, 255, 0.8);
  text-align: center;
  padding: 20px 0;
  font-size: 30px;
}
</style>
```

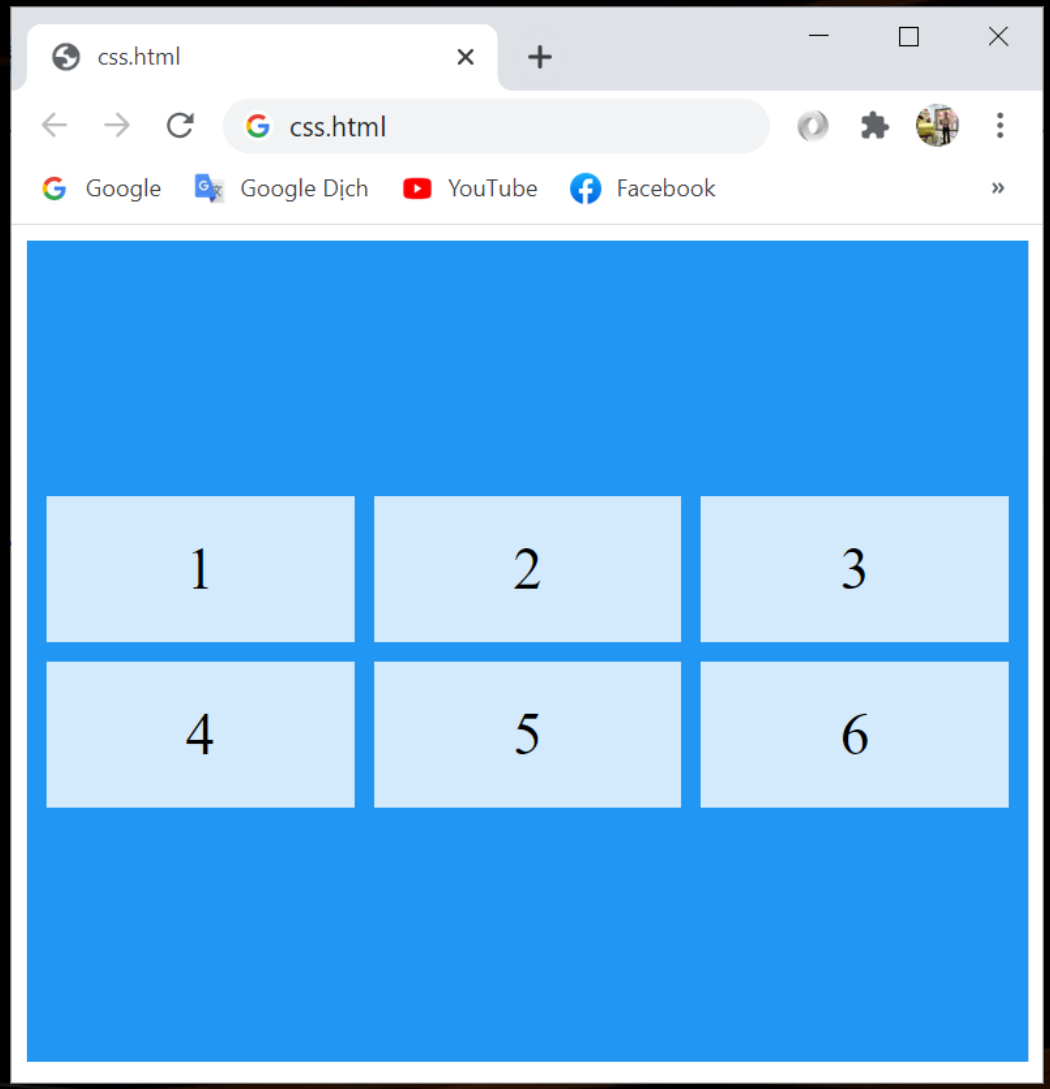
Grid – align-content – center (2)

```
<div class="grid-container">
  <div>1</div>
  <div>2</div>
  <div>3</div>
  <div>4</div>
  <div>5</div>
  <div>6</div>
</div>
```



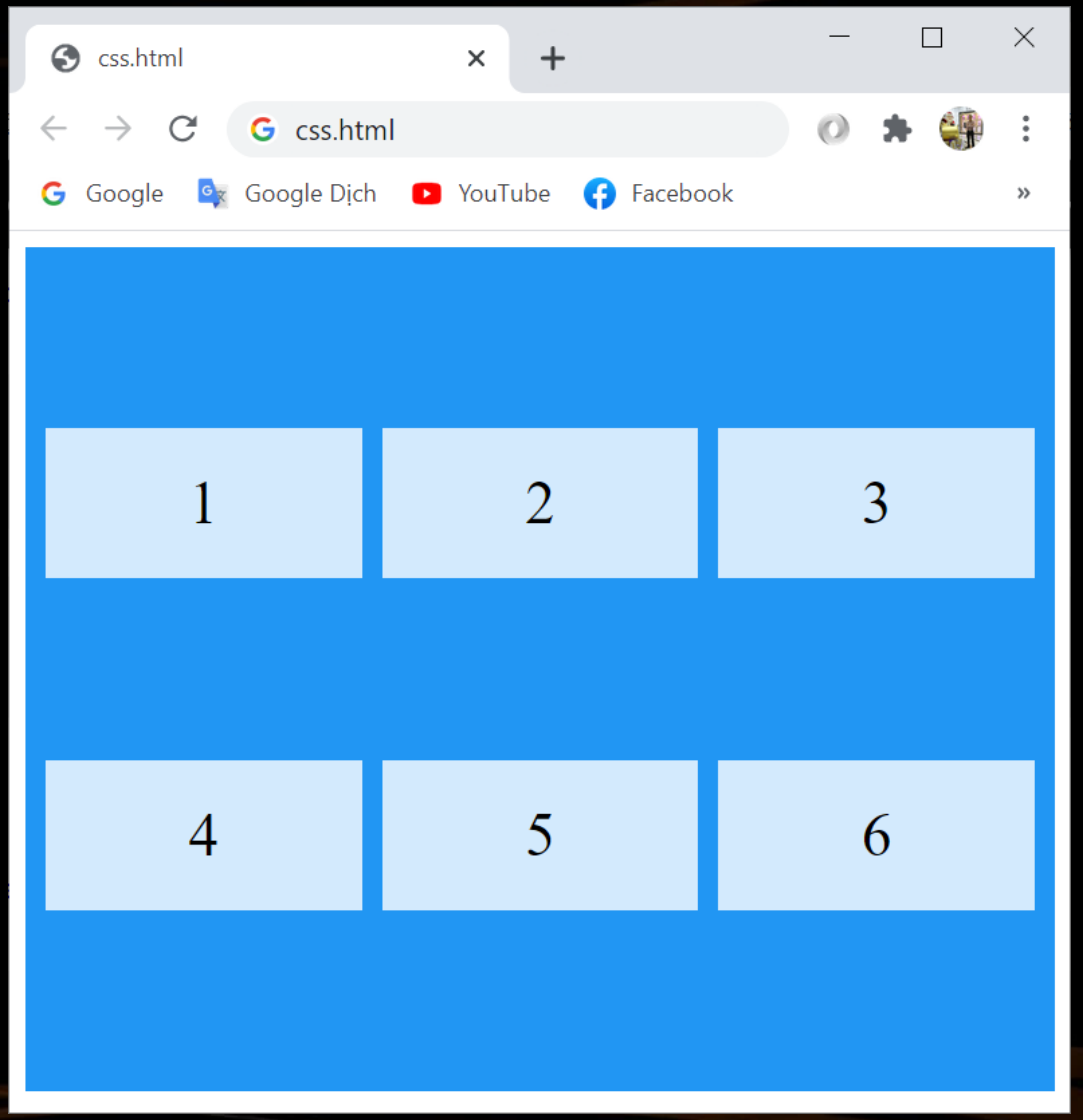
Grid – align-content – center (2)

```
<div class="grid-container">
  <div>1</div>
  <div>2</div>
  <div>3</div>
  <div>4</div>
  <div>5</div>
  <div>6</div>
</div>
```



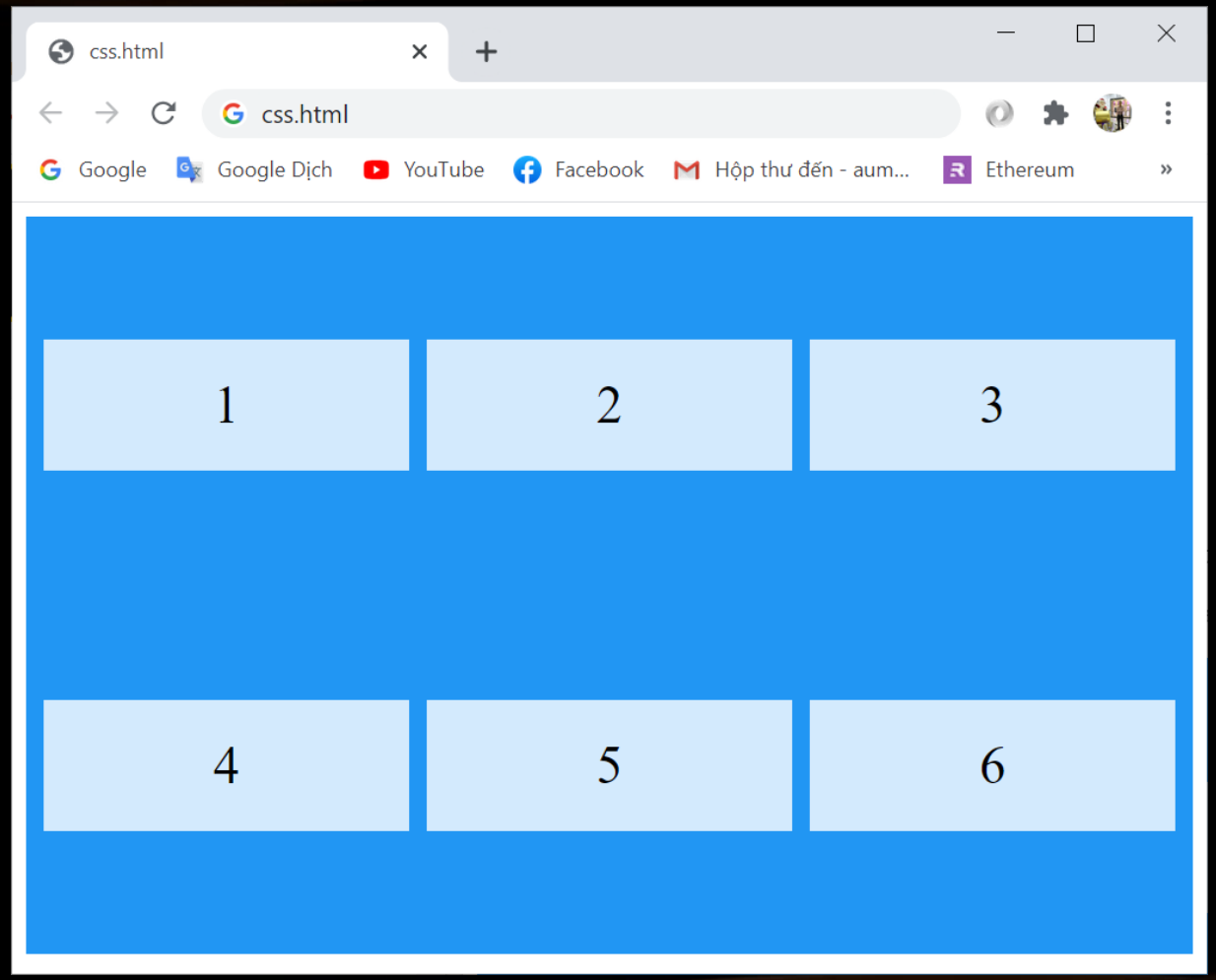
Grid – align-content – space-evenly

```
<style>
.grid-container {
    align-content: space-evenly;
}
</style>
```



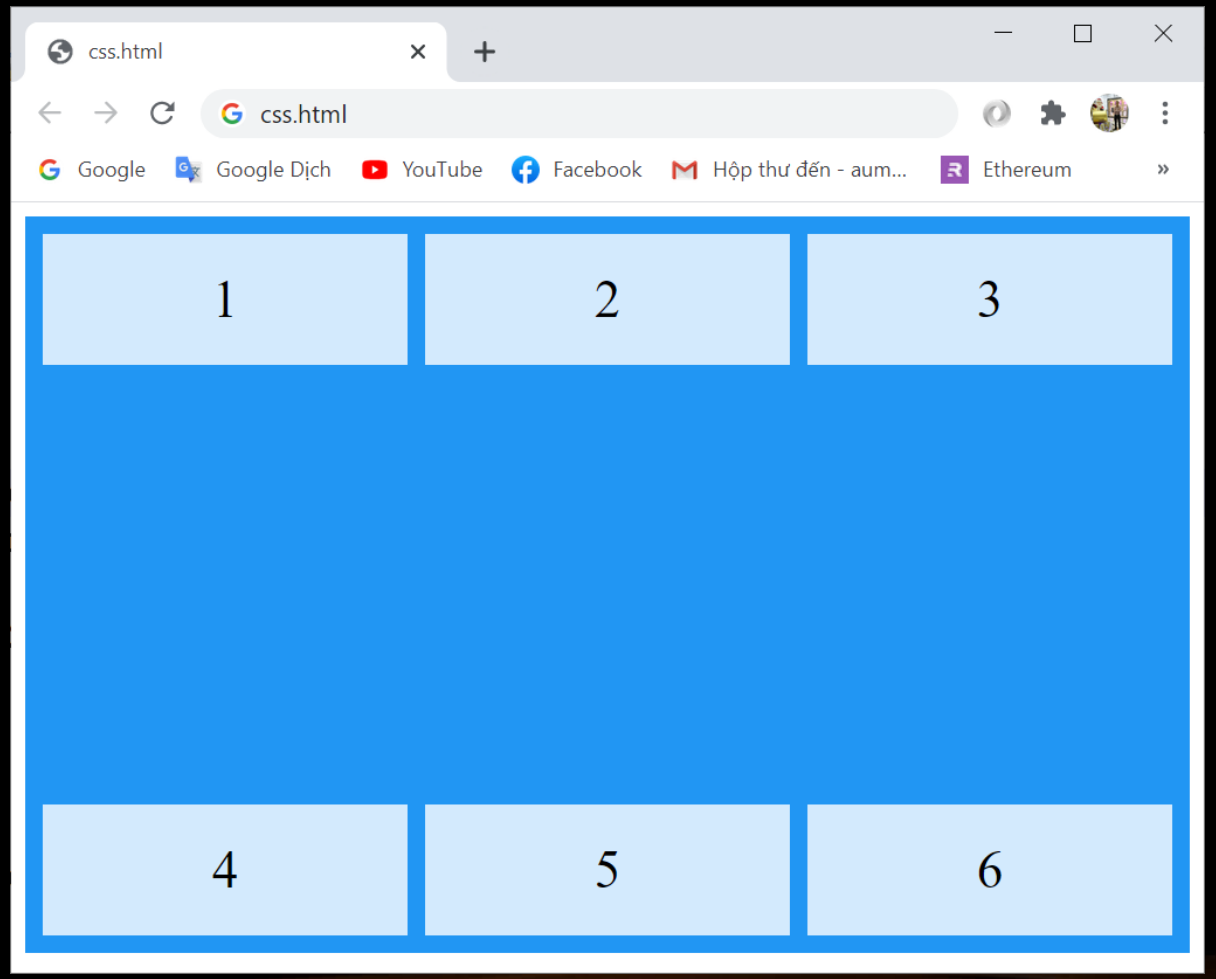
Grid – align-content – space-around

```
<style>
.grid-container {
    align-content: space-around;
}
</style>
```



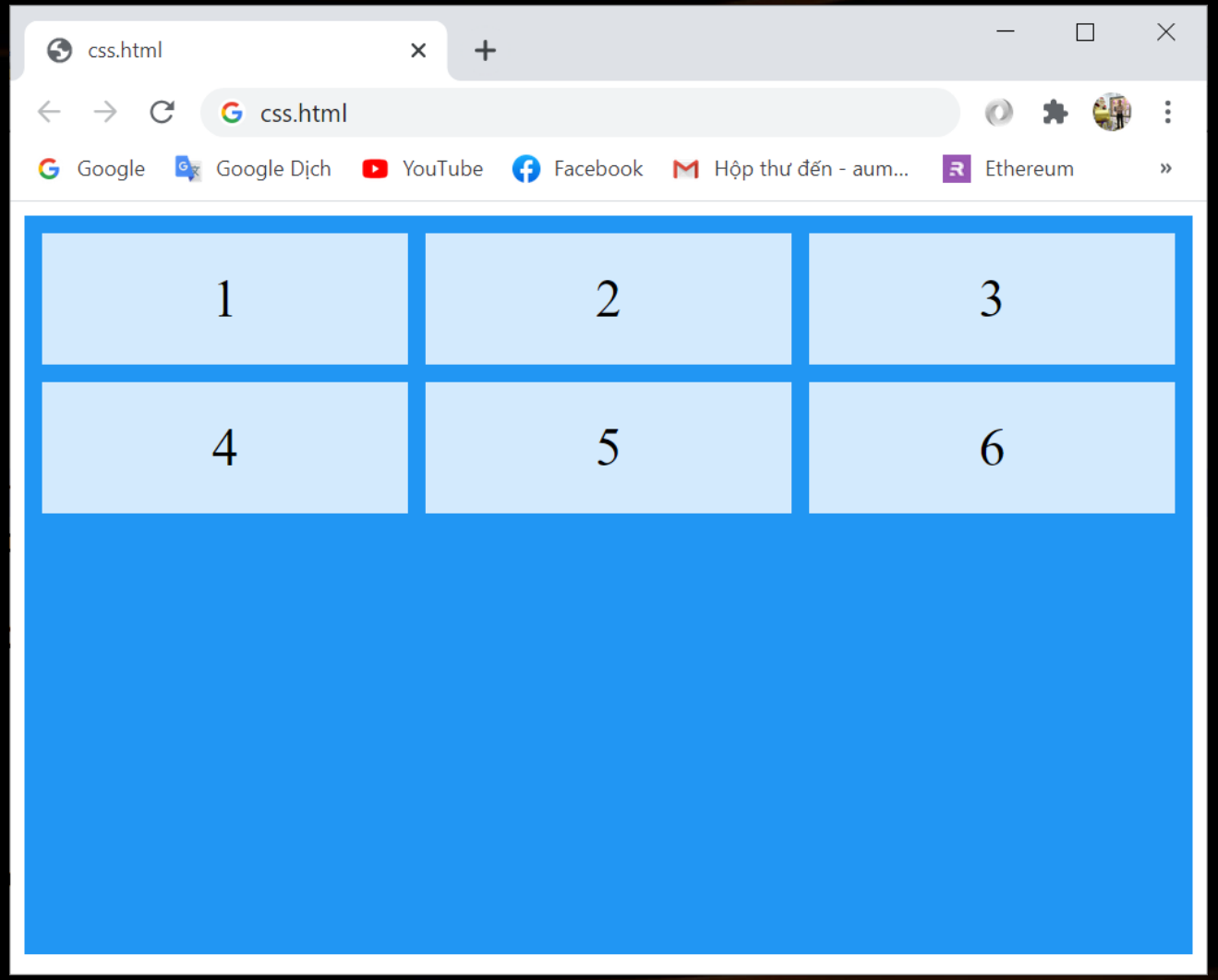
Grid – align-content – space-between

```
<style>
.grid-container {
    align-content: space-between;
}
</style>
```



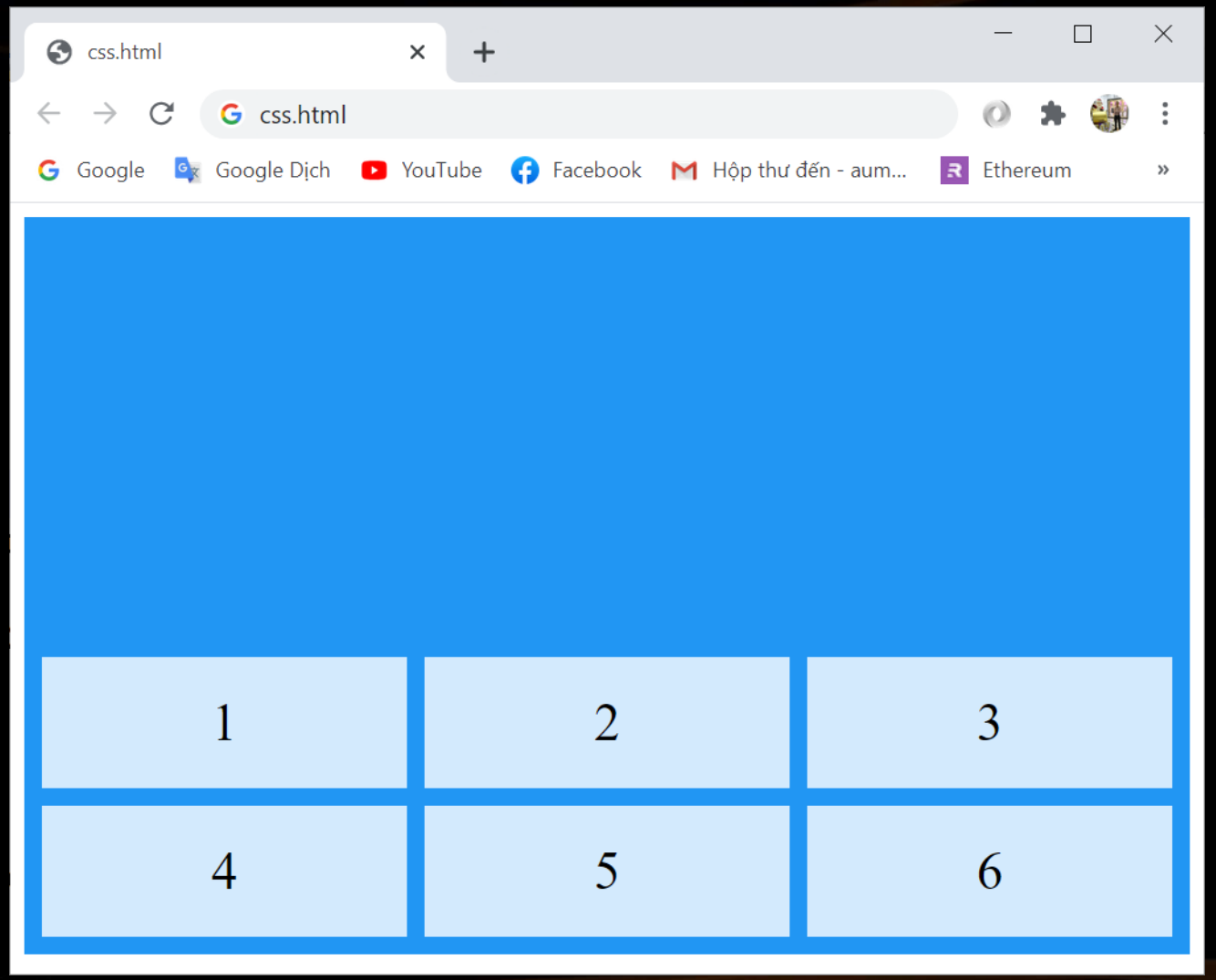
Grid – align-content – start

```
<style>
.grid-container {
    align-content: start;
}
</style>
```



Grid – align-content – end

```
<style>
.grid-container {
    align-content: end;
}
</style>
```



Grid – grid-column (1)

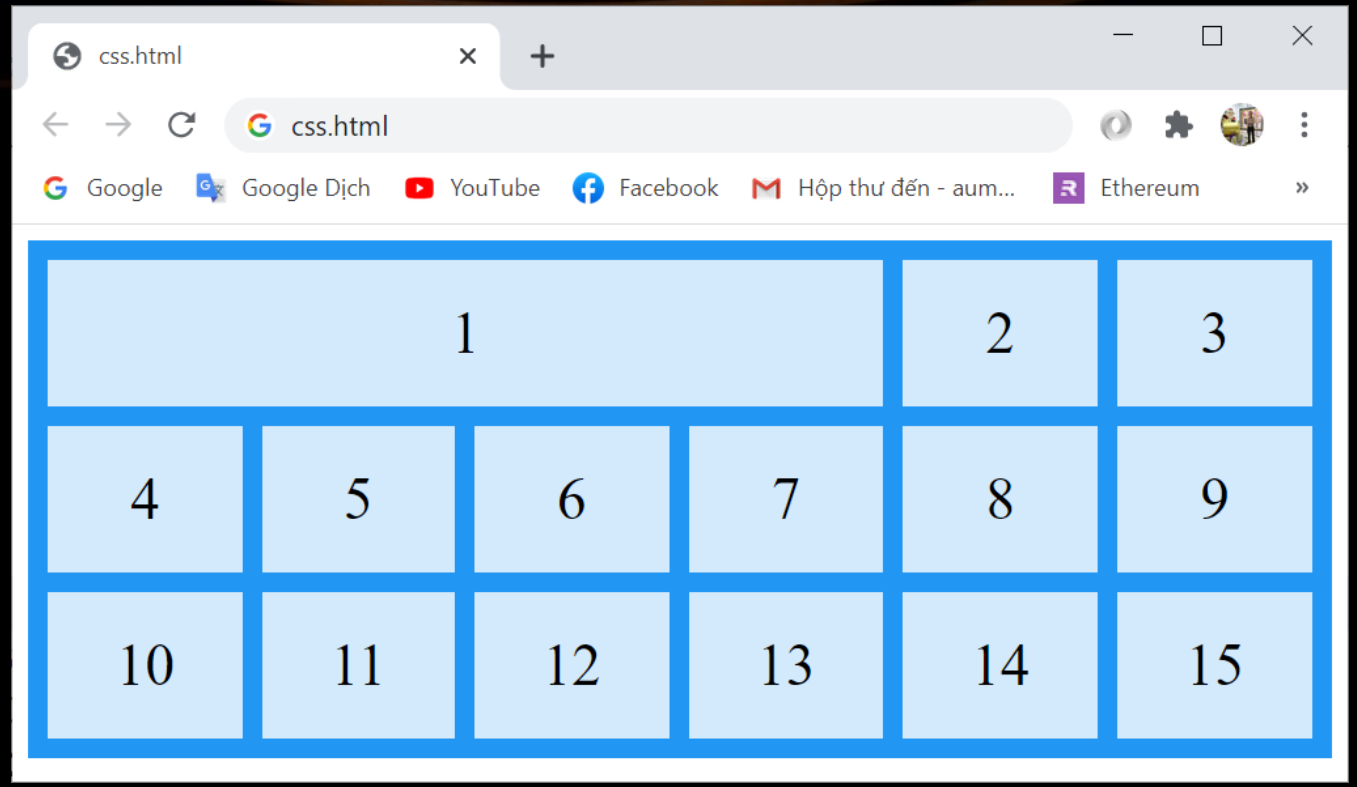
- The grid-column property defines on which column(s) to place an item

```
<style>
.grid-container {
  display: grid;
  grid-template-columns: auto auto auto auto
auto auto;
  grid-gap: 10px;
  background-color: #2196F3;
  padding: 10px;
}
.grid-container > div {
  background-color: rgba(255, 255, 255, 0.8);
  text-align: center;
  padding: 20px 0;
  font-size: 30px;
}
</style>
```

Grid – grid-column (2)

```
<style>
.item1 {
  grid-column: 1 / 5;
}
</style>
```

```
<div class="grid-container">
  <div class="item1">1</div>
  <div class="item2">2</div>
  <div class="item3">3</div>
  ...
  <div class="item14">14</div>
  <div class="item15">15</div>
  <div class="item16">16</div>
</div>
```

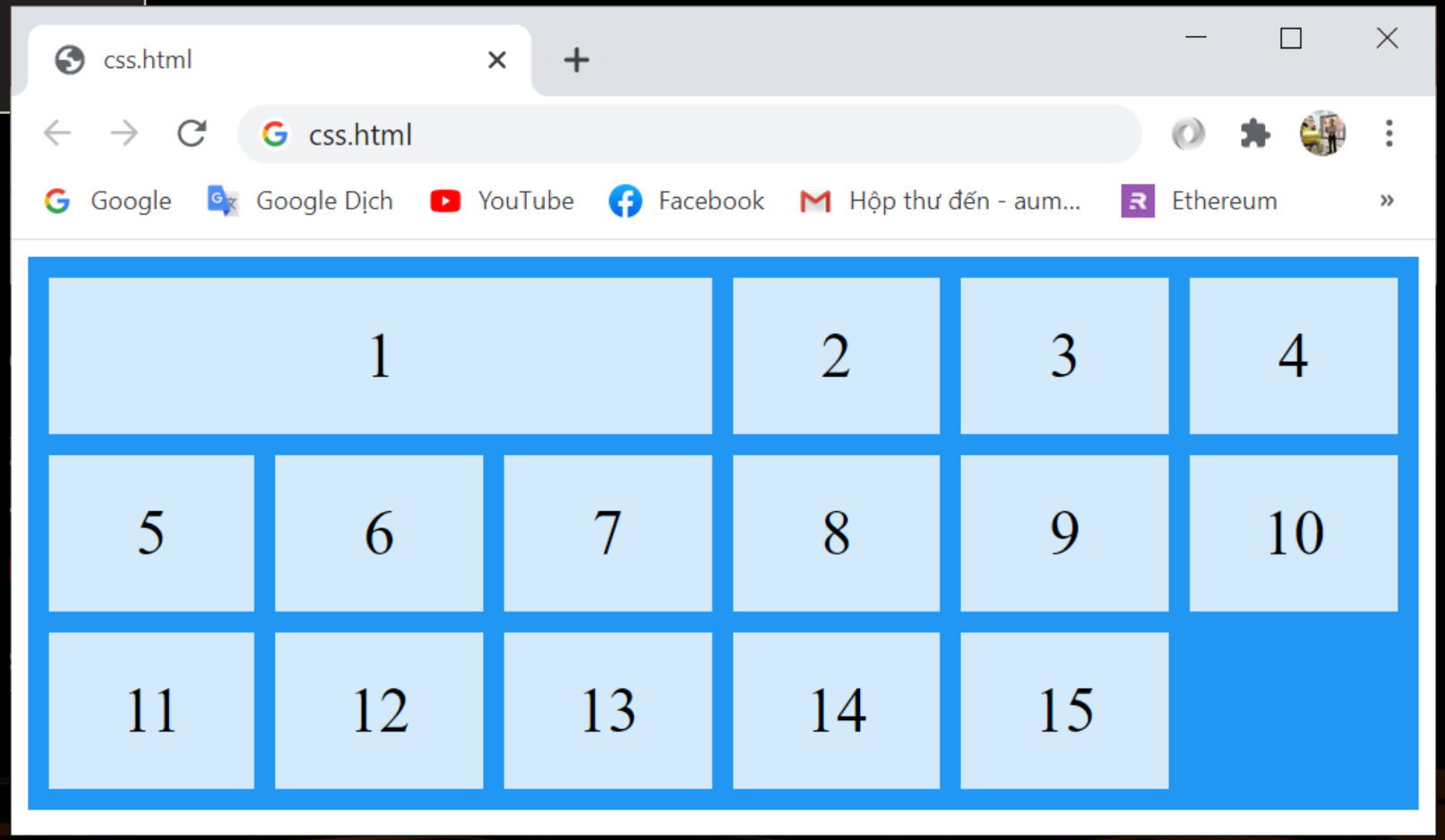


Make "item1" start on column 1 and end before column 5

Grid – grid-column (3)

```
<style>
.item1 {
  grid-column: 1 / span 3;
}
</style>
```

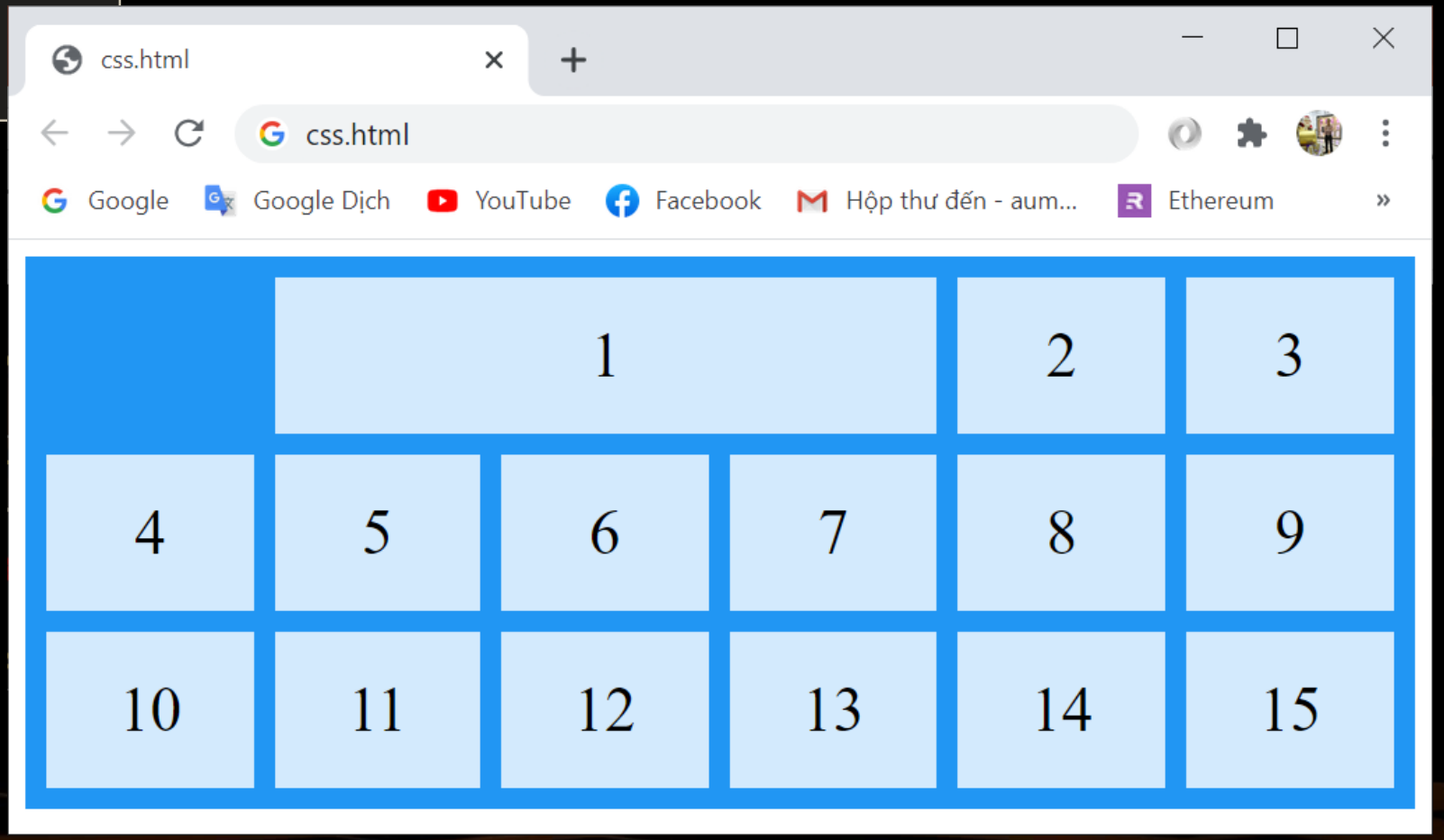
Make "item1" start on column 1 and span 3 columns



Grid – grid-column (4)

```
<style>
.item1 {
  grid-column: 2 / span 3;
}
</style>
```

Make "item2" start on column 2 and span 3 columns



Grid – grid-row (1)

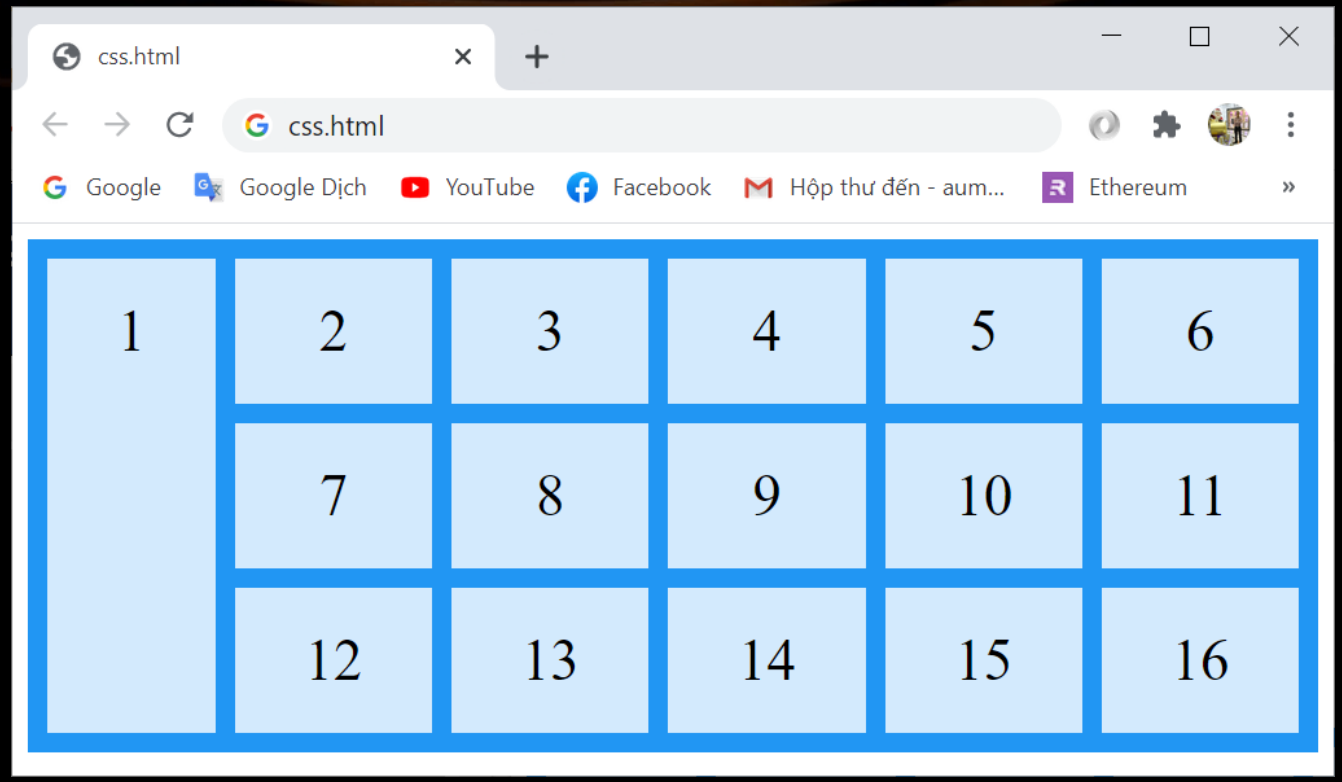
- The grid-row property defines on which row to place an item

```
<style>
.grid-container {
  display: grid;
  grid-template-columns: auto auto auto auto
auto auto;
  grid-gap: 10px;
  background-color: #2196F3;
  padding: 10px;
}
.grid-container > div {
  background-color: rgba(255, 255, 255, 0.8);
  text-align: center;
  padding: 20px 0;
  font-size: 30px;
}
</style>
```

Grid – grid-row (2)

```
<style>
.item1 {
  grid-row: 1 / 4;
}
</style>
```

```
<div class="grid-container">
  <div class="item1">1</div>
  <div class="item2">2</div>
  <div class="item3">3</div>
  ...
  <div class="item14">14</div>
  <div class="item15">15</div>
  <div class="item16">16</div>
</div>
```

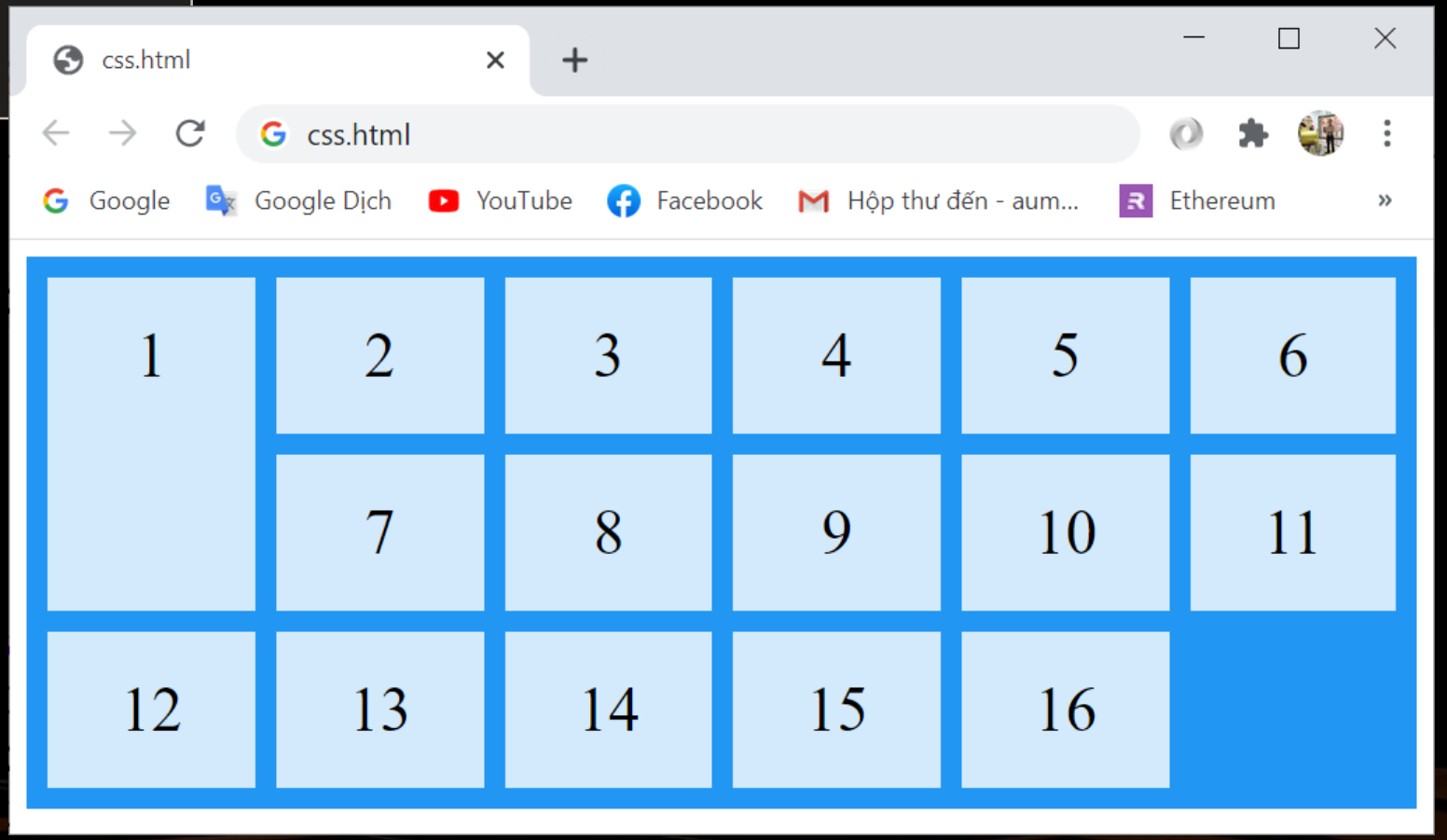


Make "item1" start on row-line 1 and end on row-line 4

Grid – grid-row (3)

```
<style>
.item1 {
  grid-row: 1 / span 2;
}
</style>
```

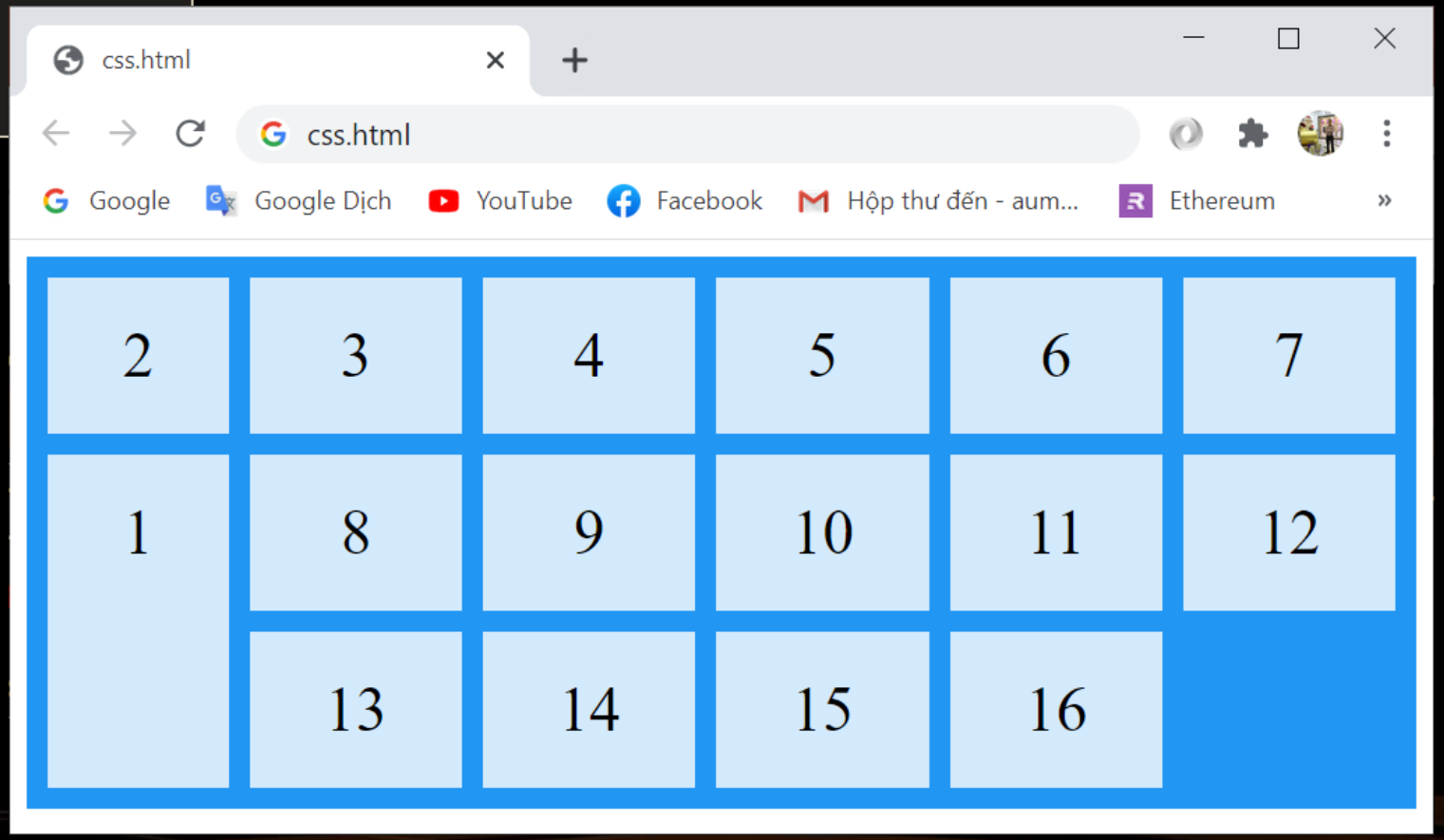
Make "item1" start on row 1 and span 2 rows



Grid – grid-row (4)

```
<style>
.item1 {
  grid-row: 2 / span 2;
}
</style>
```

Make "item1" start on row 2 and span 2 rows



Grid – grid-area (1)

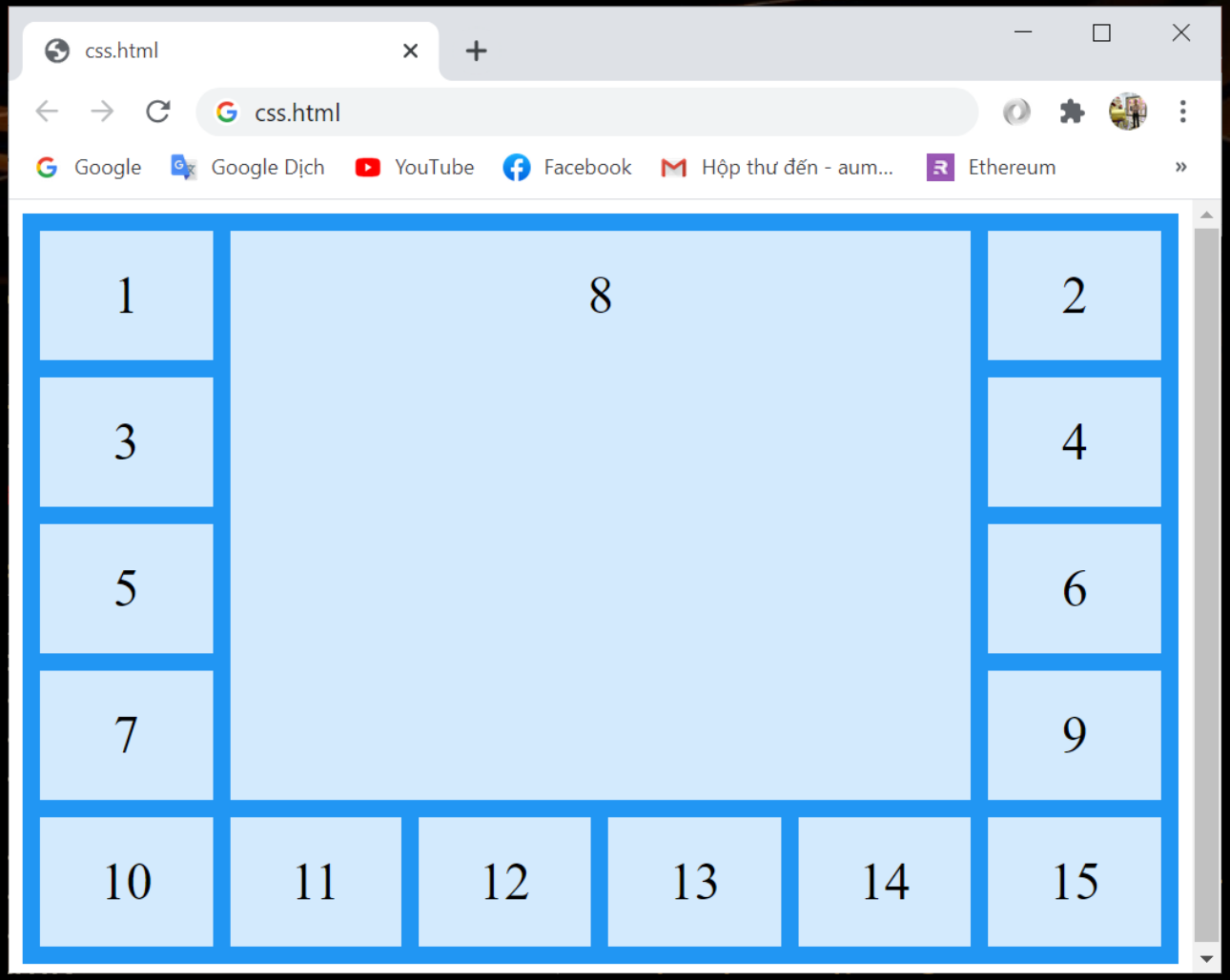
- The grid-area property can be used as a shorthand property for the grid-row-start, grid-column-start, grid-row-end and the grid-column-end properties

```
<style>
.grid-container {
  display: grid;
  grid-template-columns: auto auto auto auto
auto auto;
  grid-gap: 10px;
  background-color: #2196F3;
  padding: 10px;
}
.grid-container > div {
  background-color: rgba(255, 255, 255, 0.8);
  text-align: center;
  padding: 20px 0;
  font-size: 30px;
}
</style>
```

Grid – grid-area (2)

```
<style>
.item8 {
  grid-area: 1 / 2 / 5 / 6;
}
</style>
```

```
<div class="grid-container">
  <div class="item1">1</div>
  <div class="item2">2</div>
  <div class="item3">3</div>
  ...
  <div class="item13">13</div>
  <div class="item14">14</div>
  <div class="item15">15</div>
</div>
```

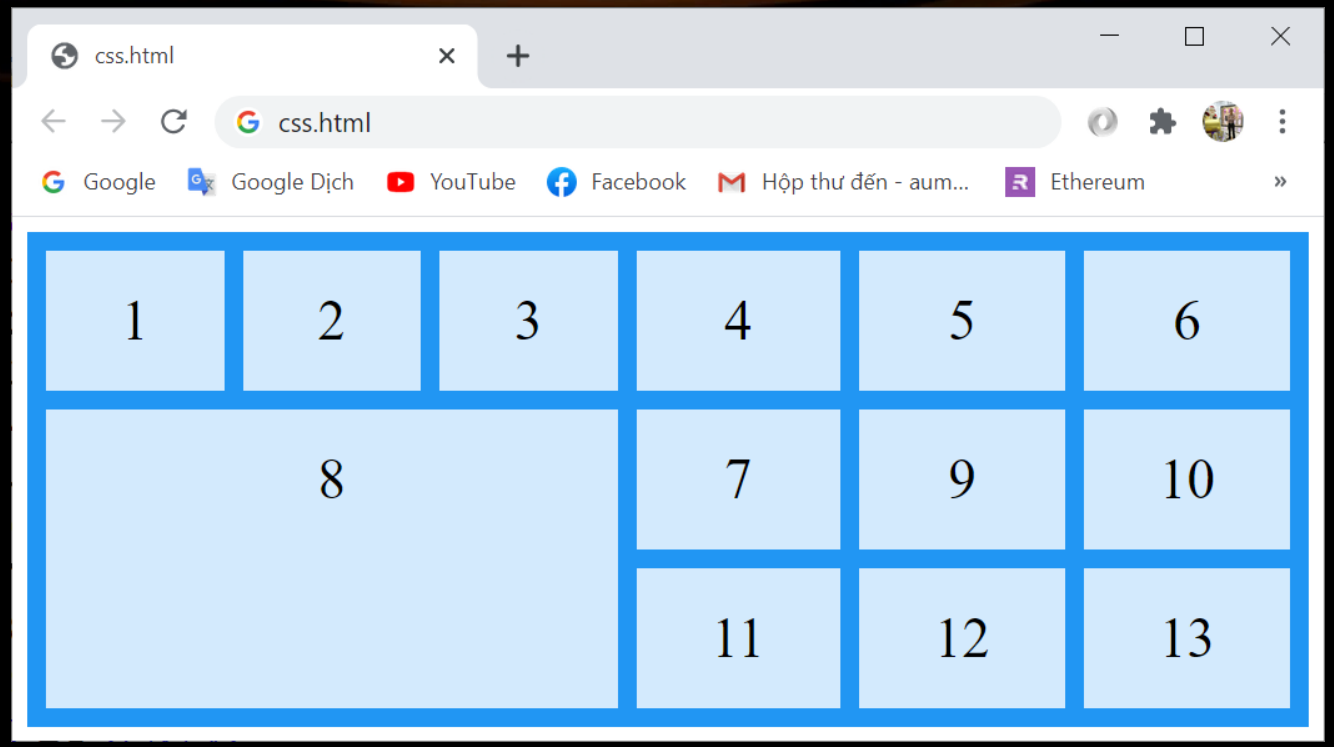


Make "item8" start on row-line 1 and column-line 2, and end on row-line 5 and column line 6

Grid – grid-area (3)

```
<style>
.item8 {
  grid-area: 2 / 1 / span 2 /
span 3;
}
</style>
```

```
<div class="grid-container">
  <div class="item1">1</div>
  <div class="item2">2</div>
  <div class="item3">3</div>
  ...
  <div class="item11">11</div>
  <div class="item12">12</div>
  <div class="item13">13</div>
</div>
```



Make "item8" start on row-line 2 and column-line 1, and span 2 rows and 3 columns

Grid – Naming Grid Items (1)

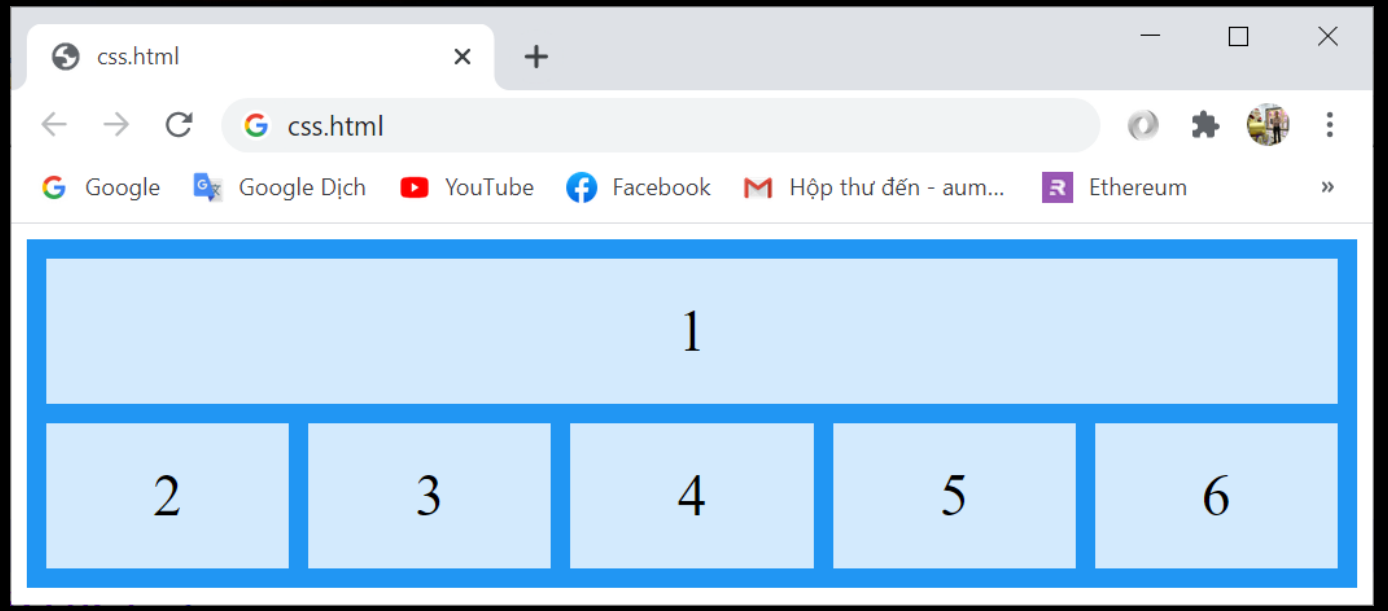
- The grid-area property can also be used to assign names to grid items

```
<style>
.item1 {
  grid-area: myArea;
}
</style>
```

```
<style>
.grid-container {
  display: grid;
  grid-template-areas: 'myArea myArea myArea myArea myArea myArea';
  grid-gap: 10px;
  background-color: #2196F3;
  padding: 10px;
}
.grid-container > div {
  background-color: rgba(255, 255, 255, 0.8);
  text-align: center;
  padding: 20px 0;
  font-size: 30px;
}
</style>
```

Grid – Naming Grid Items (2)

```
<div class="grid-container">
  <div class="item1">1</div>
  <div class="item2">2</div>
  <div class="item3">3</div>
  <div class="item4">4</div>
  <div class="item5">5</div>
  <div class="item6">6</div>
</div>
```



Item1 gets the name "myArea" and spans all five columns in a five columns grid layout

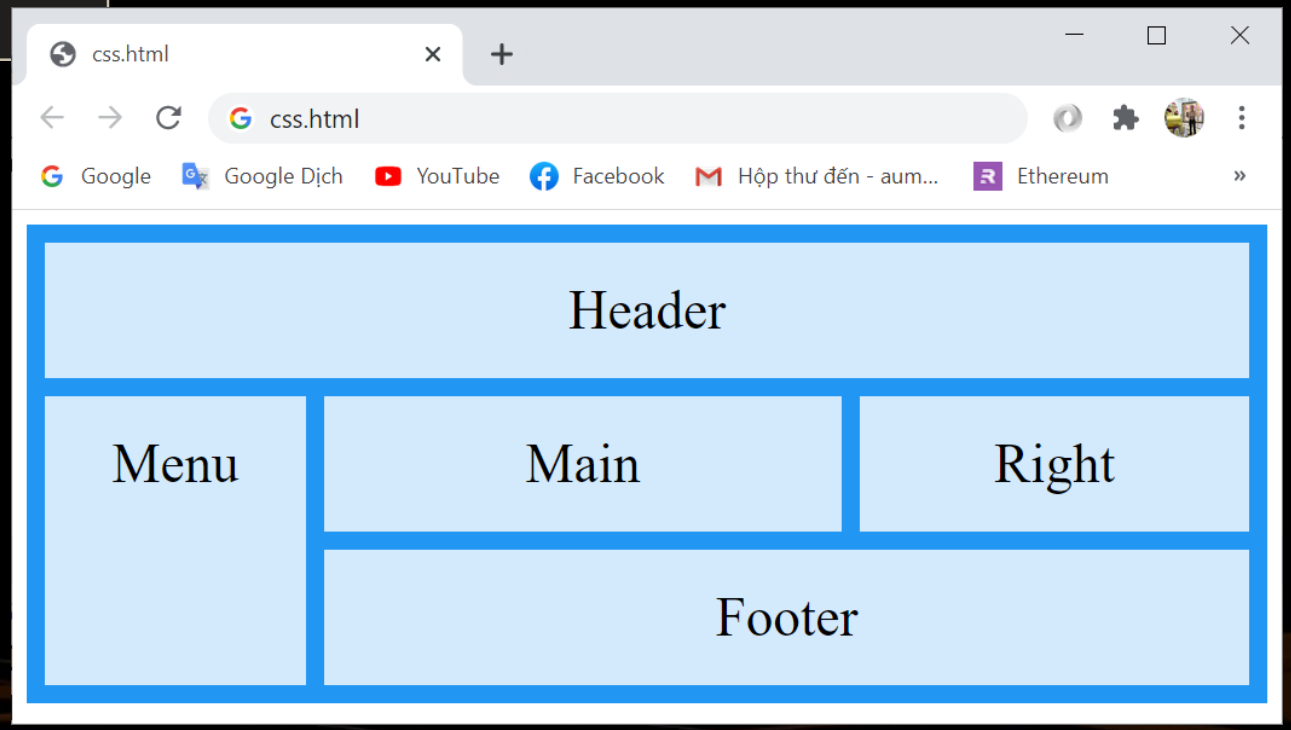
Grid – Naming Grid Items (3)

```
<style>
.grid-container {
  display: grid;
  grid-template-areas:
    'header header header header header header'
    'menu main main main right right'
    'menu footer footer footer footer footer';
  grid-gap: 10px;
  background-color: #2196F3;
  padding: 10px;
}
.grid-container > div {
  background-color: rgba(255, 255, 255, 0.8);
  text-align: center;
  padding: 20px 0;
  font-size: 30px;
}
```

```
.item1 {
  grid-area: header;
}
.item2 {
  grid-area: menu;
}
.item3 {
  grid-area: main;
}
.item4 {
  grid-area: right;
}
.item5 {
  grid-area: footer;
}
</style>
```

Grid – Naming Grid Items (4)

```
<div class="grid-container">
  <div class="item1">Header</div>
  <div class="item2">Menu</div>
  <div class="item3">Main</div>
  <div class="item4">Right</div>
  <div class="item5">Footer</div>
</div>
```



Grid – Order of the Items (1)

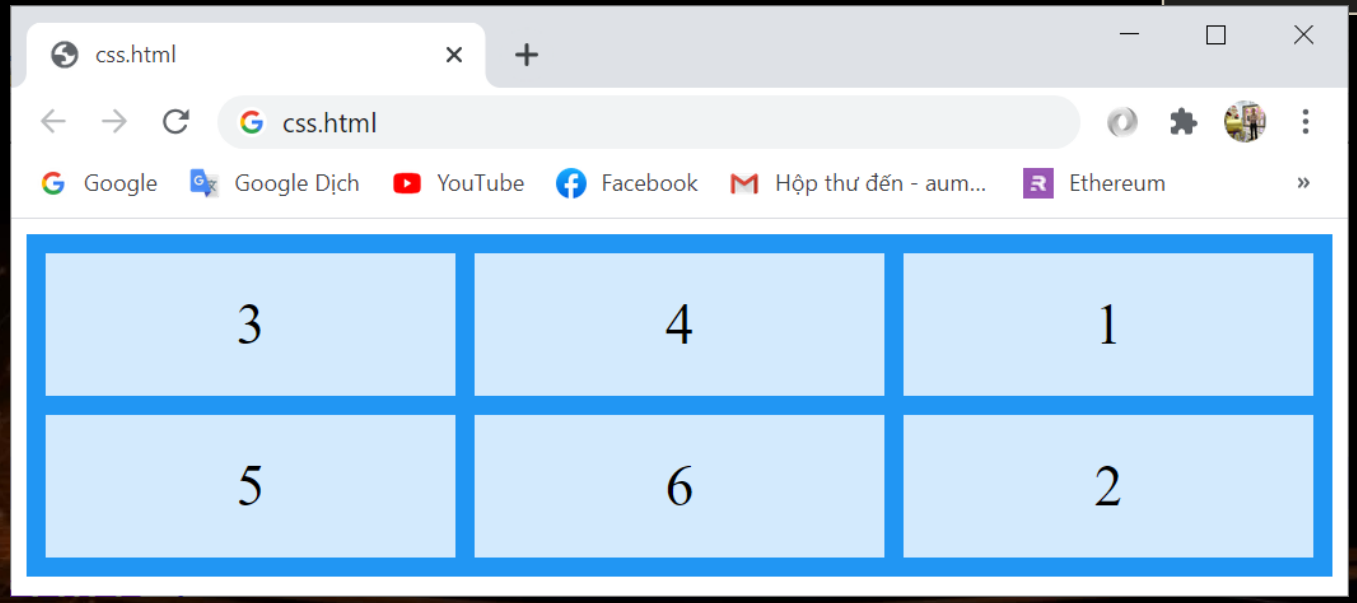
- The Grid Layout allows us to position the items anywhere we like

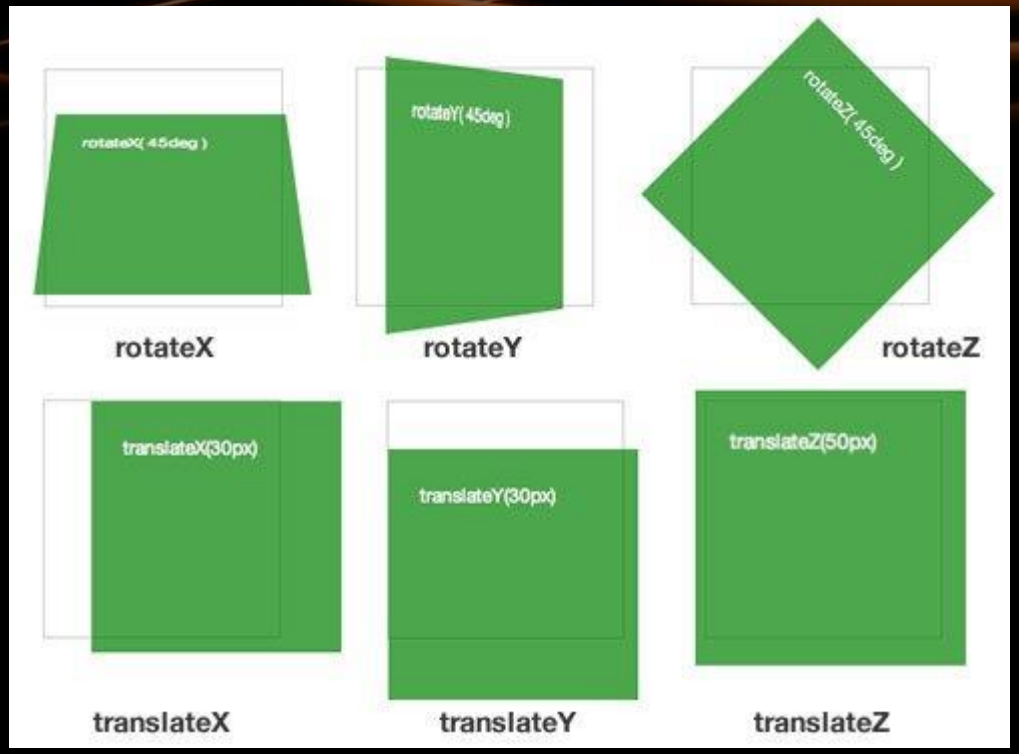
```
<style>
.grid-container {
  display: grid;
  grid-template-columns: auto auto auto;
  grid-gap: 10px;
  background-color: #2196F3;
  padding: 10px;
}
.grid-container > div {
  background-color: rgba(255, 255, 255, 0.8);
  text-align: center;
  padding: 20px 0;
  font-size: 30px;
}
</style>
```

Grid – Order of the Items (2)

```
<div class="grid-container">
  <div class="item1">Header</div>
  <div class="item2">Menu</div>
  <div class="item3">Main</div>
  <div class="item4">Right</div>
  <div class="item5">Footer</div>
</div>
```

```
<style>
.item1 { grid-area: 1 / 3 / 2 / 4; }
.item2 { grid-area: 2 / 3 / 3 / 4; }
.item3 { grid-area: 1 / 1 / 2 / 2; }
.item4 { grid-area: 1 / 2 / 2 / 3; }
.item5 { grid-area: 2 / 1 / 3 / 2; }
.item6 { grid-area: 2 / 2 / 3 / 3; }
</style>
```





CSS Transforms

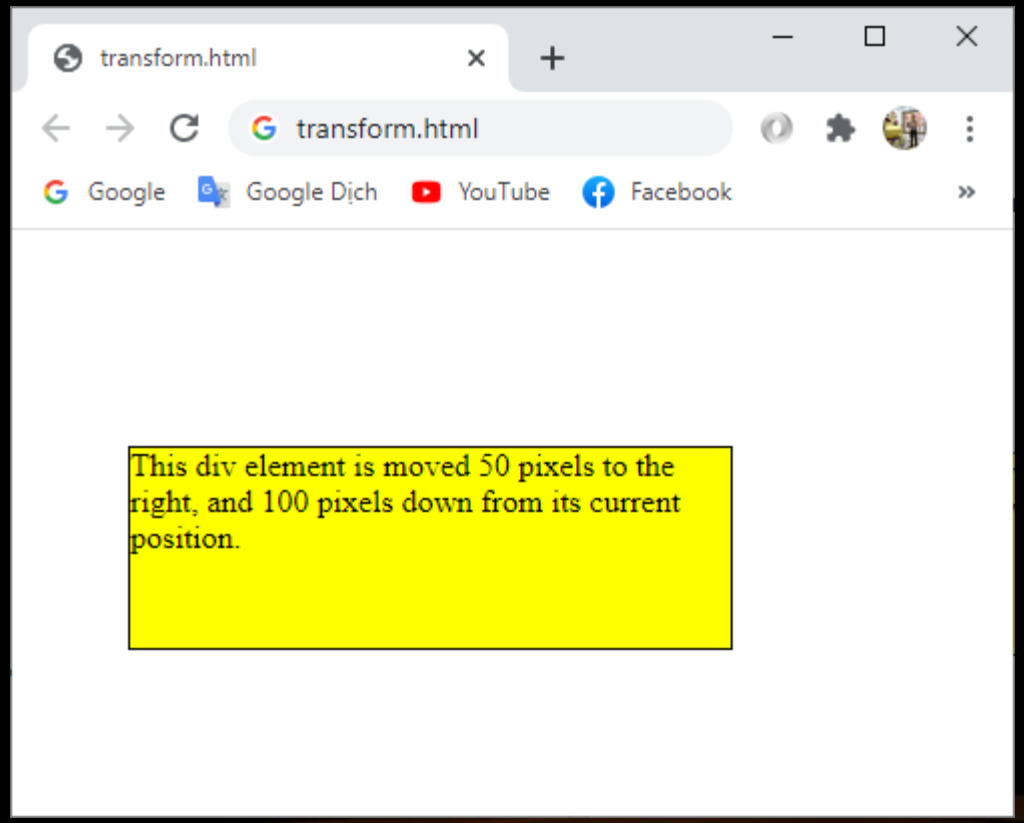
CSS 2D Transforms

- CSS transforms allow you to move, rotate, scale, and skew elements
- Key: transform
- Method:
  - translate()
  - rotate()
  - scaleX()
  - scaleY()
  - scale()
  - skewX()
  - skewY()
  - skew()
  - matrix()

CSS 2D Transforms – translate()

```
<div>
This div element is moved 50 pixels to the right, and 100 pixels down from its current position.
</div>
```

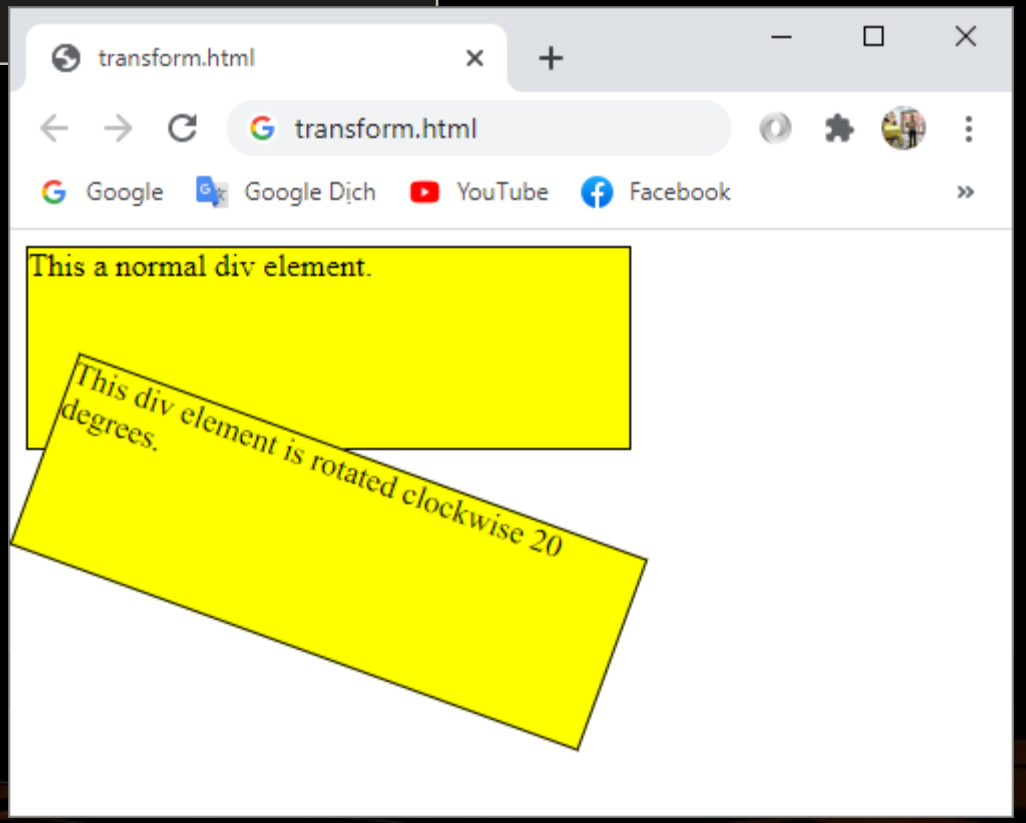
```
<style>
div {
  width: 300px;
  height: 100px;
  background-color: yellow;
  border: 1px solid black;
transform: translate(50px,100px);
}
</style>
```



CSS 2D Transforms – rotate()

```
<div>
This a normal div element.
</div>
<div id="myDiv">
This div element is rotated clockwise 20 degrees.
</div>
```

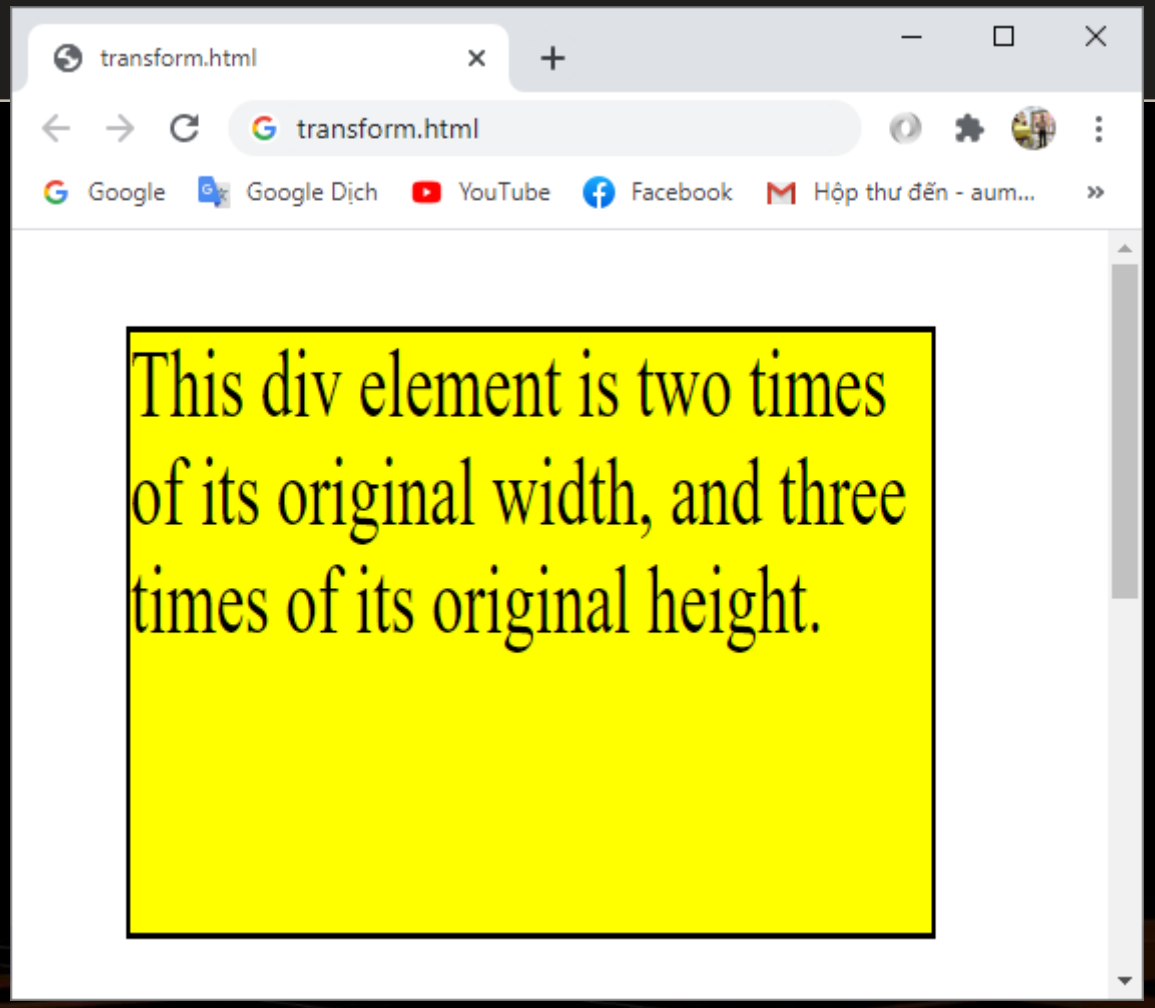
```
<style>
div {
  width: 300px;
  height: 100px;
  background-color: yellow;
  border: 1px solid black;
}
div#myDiv {
  transform: rotate(20deg);
}</style>
```



CSS 2D Transforms – scale()

```
<div>
This div element is two times of its original width, and three times of its original height.
</div>
```

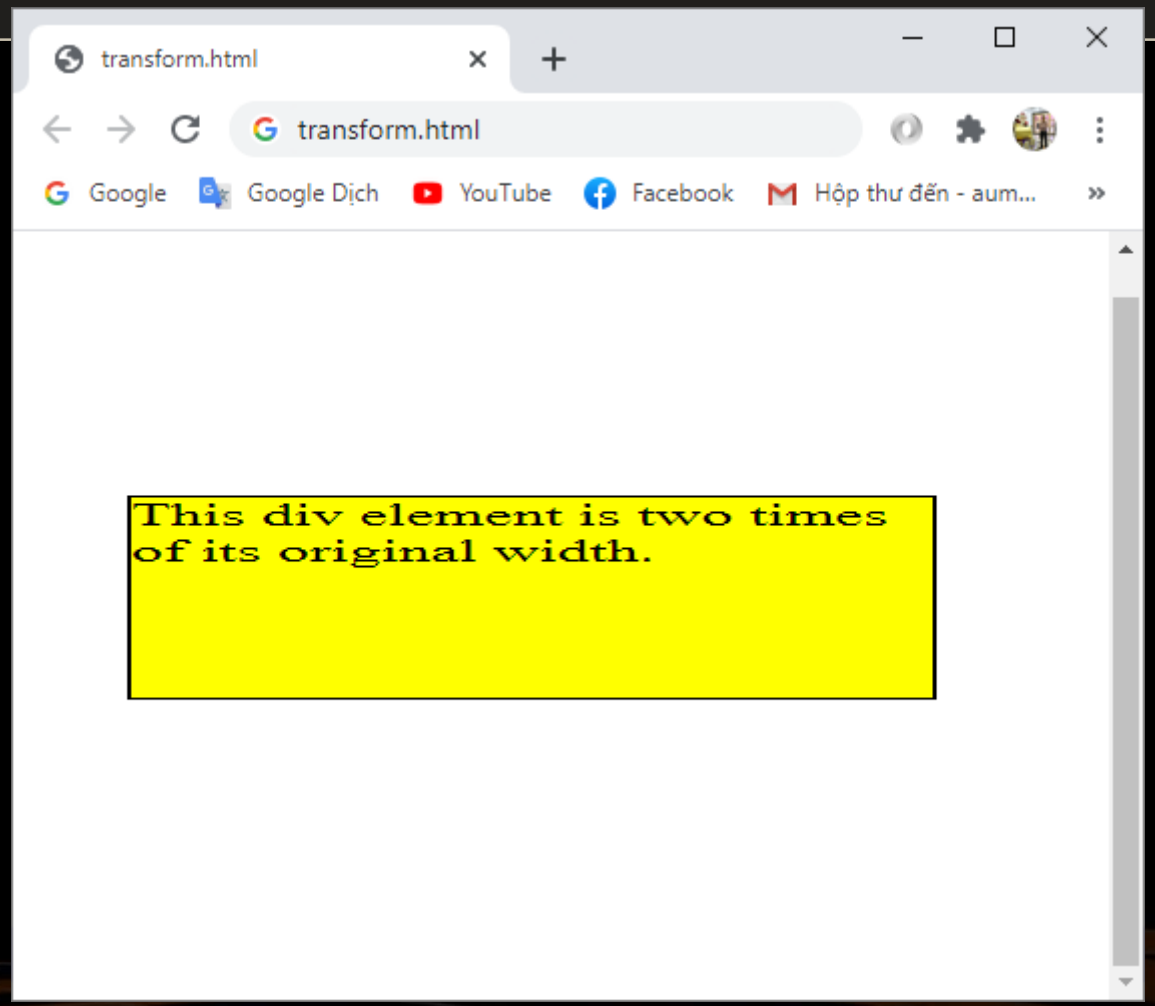
```
<style>
div {
  margin: 150px;
  width: 200px;
  height: 100px;
  background-color: yellow;
  border: 1px solid black;
  transform: scale(2,3);
}
</style>
```



CSS 2D Transforms – scaleX()

```
<div>
This div element is two times of its original width.
</div>
```

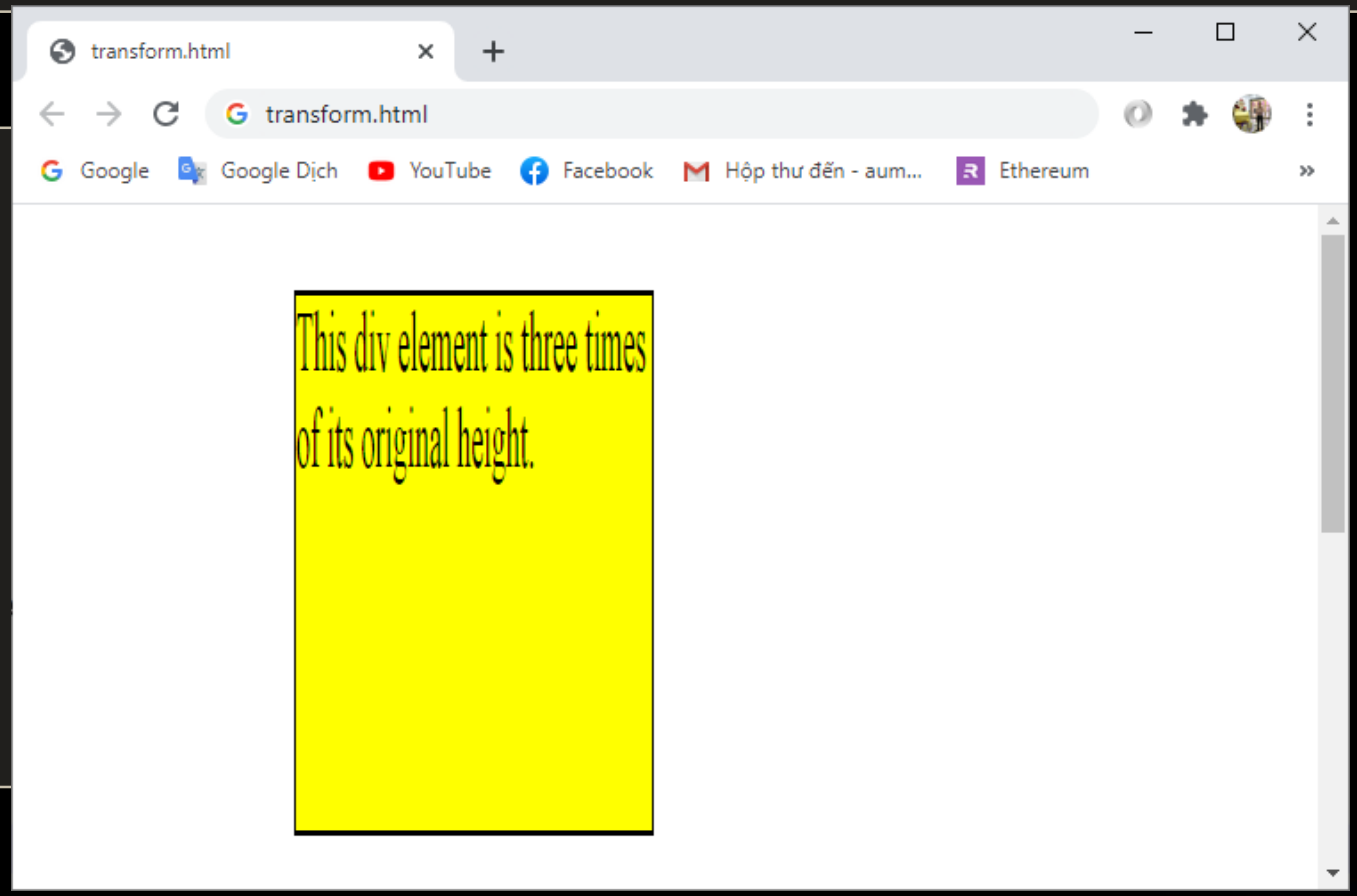
```
<style>
div {
 margin: 150px;
  width: 200px;
  height: 100px;
  background-color: yellow;
  border: 1px solid black;
  transform: scaleX(2);
}
</style>
```



CSS 2D Transforms – scaleY()

```
<div>
This div element is three times of its original height.
</div>
```

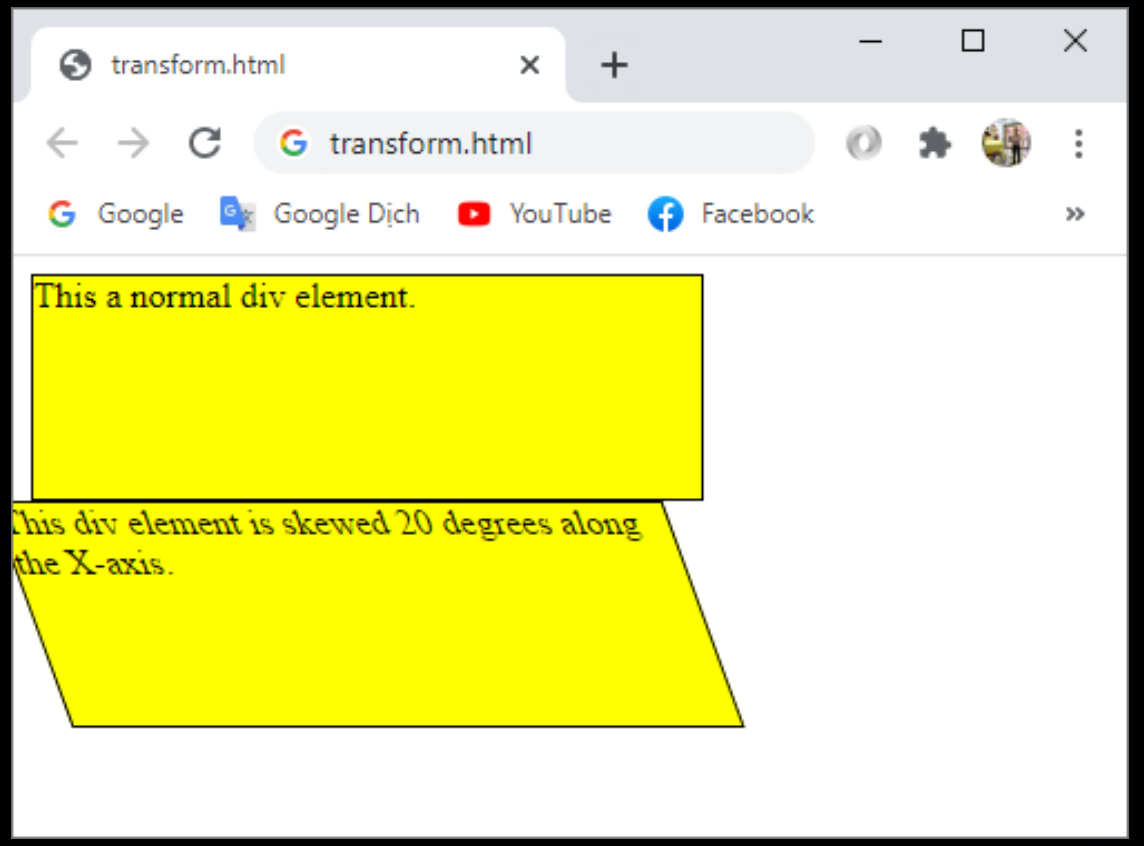
```
<style>
div {
  margin: 150px;
  width: 200px;
  height: 100px;
  background-color: yellow;
  border: 1px solid black;
  transform: scaleY(3);
</style>
```



CSS 2D Transforms – skewX()

```
<div>This a normal div element.</div>
<div id="myDiv">
This div element is skewed 20 degrees along the X-axis.</div>
```

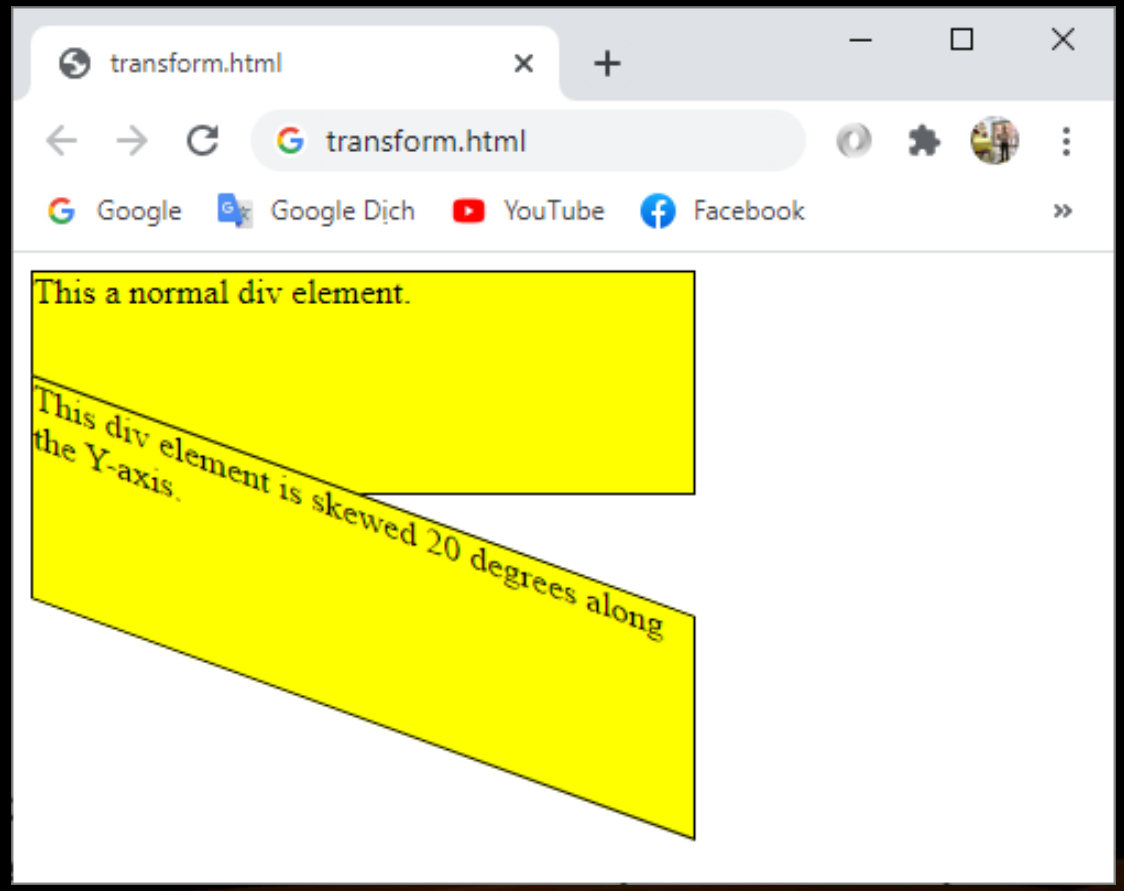
```
<style>
div {
  width: 300px;
  height: 100px;
  background-color: yellow;
  border: 1px solid black;
}
div#myDiv {
  transform: skewX(20deg);
}
</style>
```



CSS 2D Transforms – skewY()

```
<div>This a normal div element.</div>
<div id="myDiv">This div element is skewed 20 degrees along the Y-axis.</div>
```

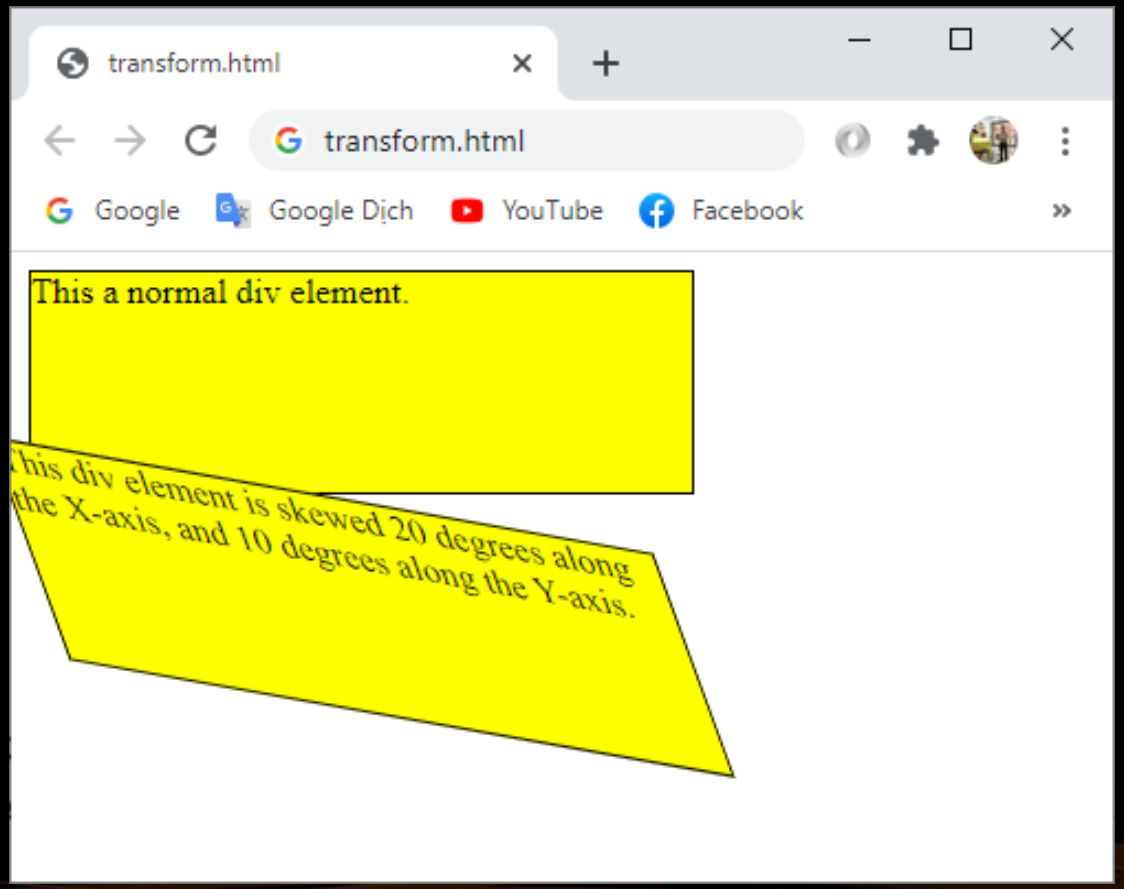
```
<style>
div {
  width: 300px;
  height: 100px;
  background-color: yellow;
  border: 1px solid black;
}
div#myDiv {
  transform: skewY(20deg);
}
</style>
```



CSS 2D Transforms – skew()

```
<div>This a normal div element.</div>
<div id="myDiv">This div element is skewed 20 degrees along the X-axis, and
10 degrees along the Y-axis.</div>
```

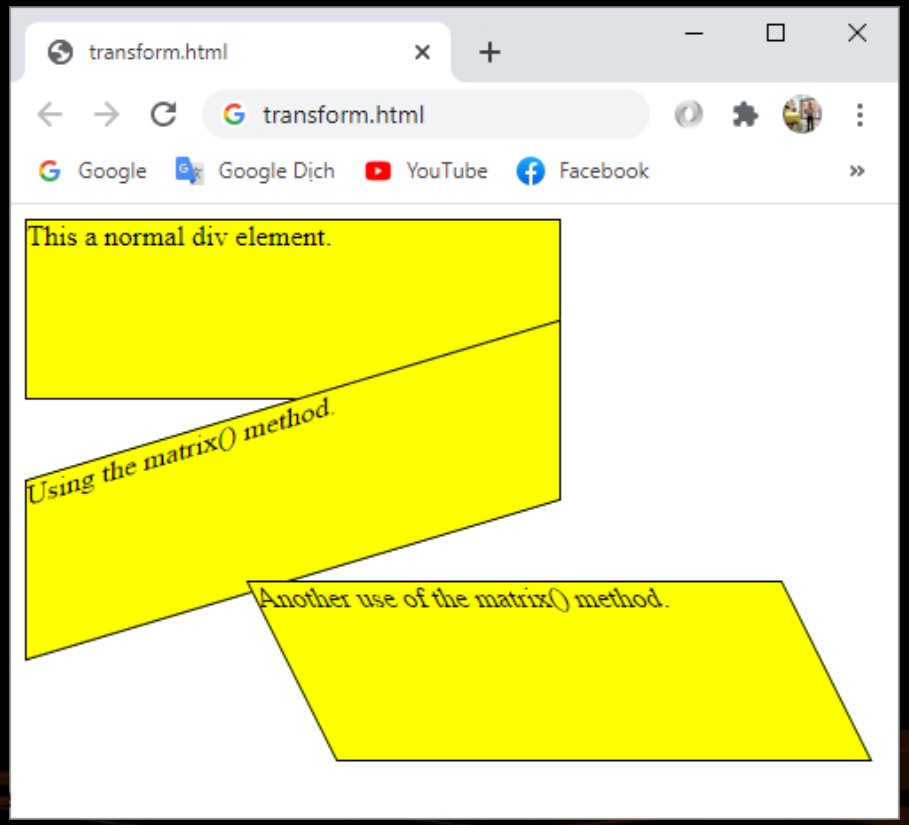
```
<style>
div {
  width: 300px;
  height: 100px;
  background-color: yellow;
  border: 1px solid black;
}
div#myDiv {
  transform: skew(20deg,10deg);
}
</style>
```



CSS 2D Transforms – matrix()

```
<style>
div {
  width: 300px;
  height: 100px;
  background-color:
yellow;
  border: 1px solid black;
}
div#myDiv1 {
  transform: matrix(1, -
0.3, 0, 1, 0, 0);
}
div#myDiv2 {
  transform: matrix(1, 0,
0.5, 1, 150, 0);
}
</style>
```

```
<div>This a normal div element.</div>
<div id="myDiv1">Using the matrix() method.
</div>
<div id="myDiv2">Another use of the matrix()
method.</div>
```



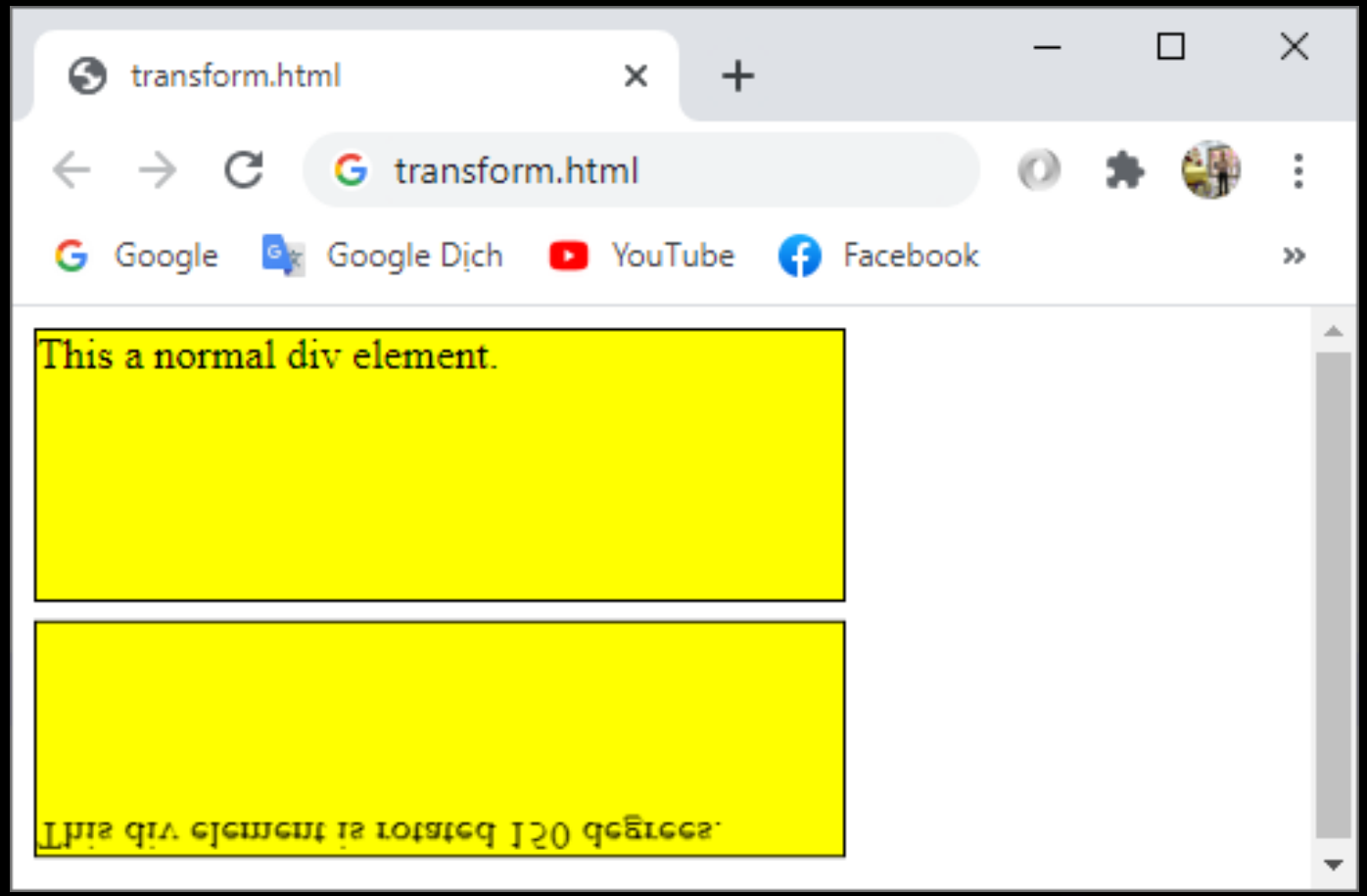
CSS 3D Transforms

- CSS also supports 3D transformations
- Key: transform
- Method:
  - rotateX()
  - rotateY()
  - rotateZ()

CSS 3D Transforms – rotateX()

```
<div>This a normal div element.</div>
<div id="myDiv">This div element is rotated 150 degrees. </div>
```

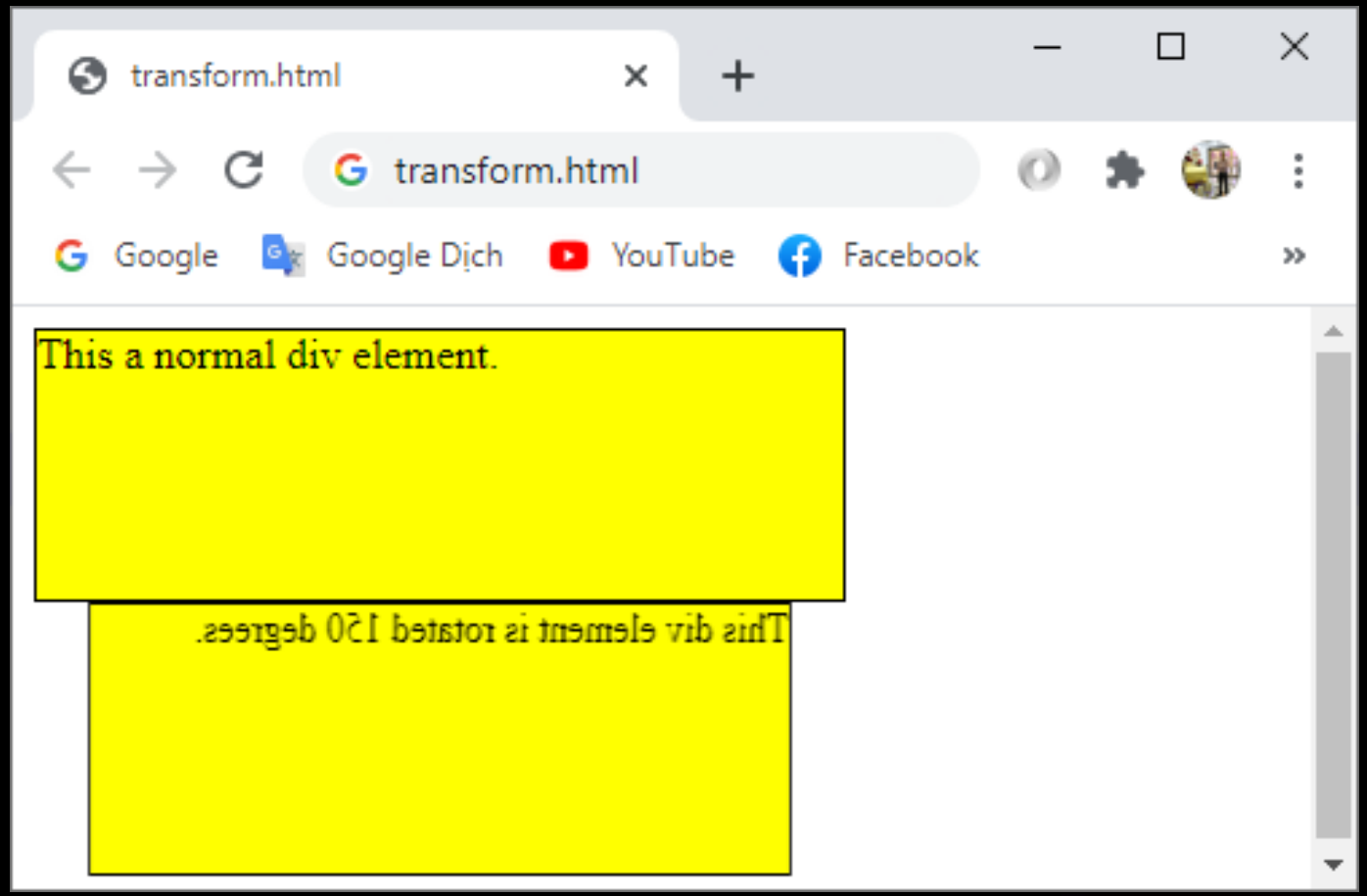
```
<style>
div {
  width: 300px;
  height: 100px;
  background-color: yellow;
  border: 1px solid black;
}
#myDiv {
  transform: rotateX(150deg);
}
</style>
```



CSS 3D Transforms – rotateY()

```
<div>This a normal div element.</div>
<div id="myDiv">This div element is rotated 150 degrees. </div>
```

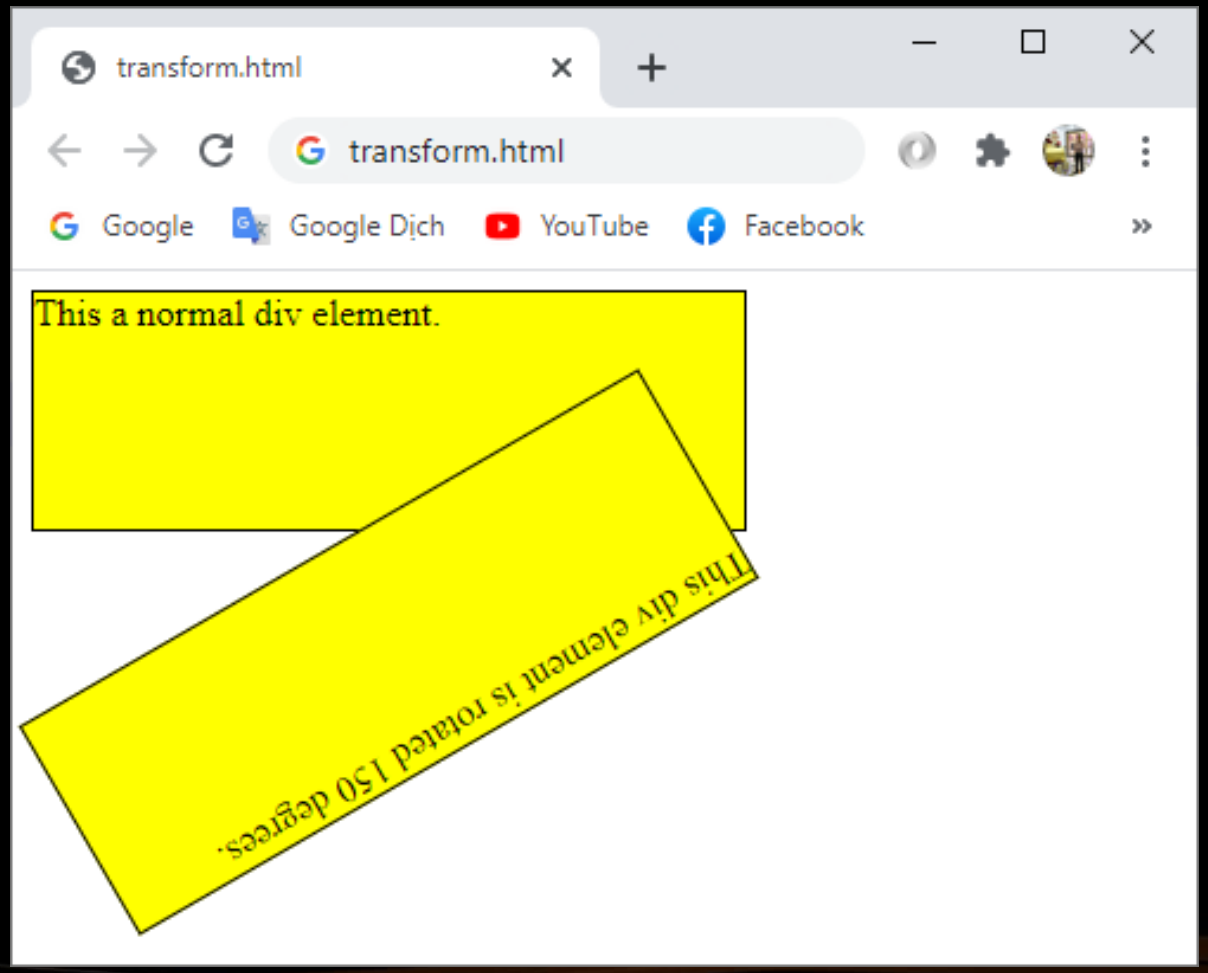
```
<style>
div {
  width: 300px;
  height: 100px;
  background-color: yellow;
  border: 1px solid black;
}
#myDiv {
  transform: rotateY(150deg);
}
</style>
```



CSS 3D Transforms – rotateZ()

```
<div>This a normal div element.</div>
<div id="myDiv">This div element is rotated 150 degrees. </div>
```

```
<style>
div {
  width: 300px;
  height: 100px;
  background-color: yellow;
  border: 1px solid black;
}
#myDiv {
  transform: rotateZ(150deg);
}
</style>
```



Summary

- Flexbox
  - flex-direction, flex-wrap, flex-flow, container properties
  - justify-content, align-content, align-items
- Transforms
  - CSS 2D Transforms
  - CSS 3D Transforms

CSS Layout

Questions?