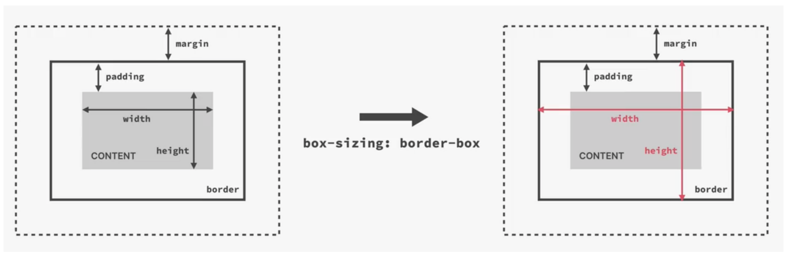
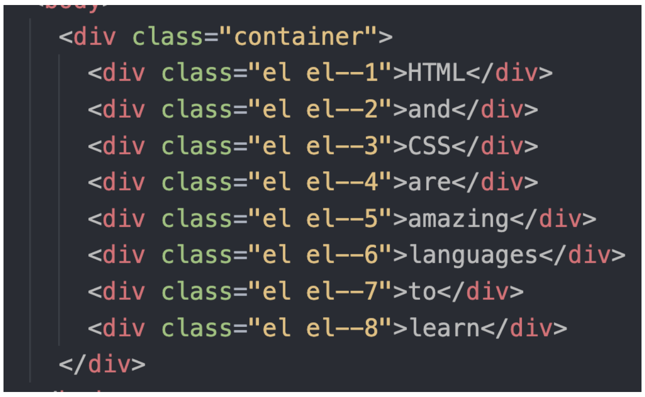
CSS-Layout

- Developer skills
  - Googling: CSS-Tricks, Stack Overflow, developer.mozilla.org
- What is Layout?
  - The way text, images, and other content is placed and arranged on a webpage
  - Give the webpage a visual structure to place the content
  - Building a Layout: not simply add elements one by one, it arranges page elements into a structure.
- Types of Layout
  - Page Layout
  - Component Layout
- Three Ways to build CSS Layout
  - ~~Floats~~
    - Outdated method,use float CSS property
  - FlexBox
    - perfert for component layouts, modern way for laying out elements in a 1-dimensional row without using floats.
  - CSS Grid
    - Fully-fledged 2-dimensional grid. Perfect for page layouts and complex componenets.
- **Float Layouts**
  - **{float:left;}** or **{float:right;}**
  - Characteristics
    - elements are removed from the normal flow:"out of flow"
    - Text and inline elements will wrap around the floated elements
    - Container will not adjust its height to the element
  - Clearing Floats
    - we add an empty <div class="clear"></div>, then write **.clear{clear:...}** in css
    - after we add a clearfix class in html element, in css **.clearfix::after{display:block;clear:...; content:"";}**   <!--add a brand new element at the very end of the container,even if the blank content is okay, but the clearfix only apply for block-->
  - **{box-sizing:border box;}**
    - The lengths of width and height get larger.



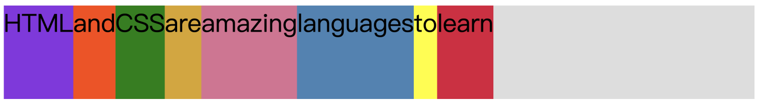
    - Final element width=width; Final element height=height
    - usually, it will add into universial element if we want to apply this for every single element in this page

- **FlexBox**
  - set a container element with many child elements

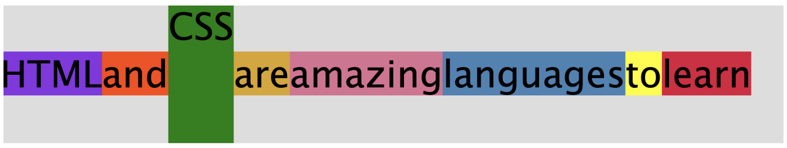


```
  <div class="container">
    <div class="el el--1">HTML</div>
    <div class="el el--2">and</div>
    <div class="el el--3">CSS</div>
    <div class="el el--4">are</div>
    <div class="el el--5">amazing</div>
    <div class="el el--6">languages</div>
    <div class="el el--7">to</div>
    <div class="el el--8">learn</div>
  </div>
```

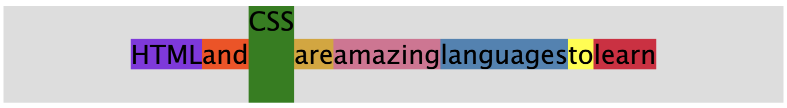
  - **{display:flex;}** In webpage, the contents are immediately side-by-side. These elements here we called them--flex item



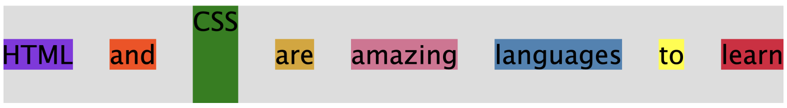
  - vertically centering
    - **{align-items:center;}** in the container



    - By default, align-items is set to be **stretch** as the tallest elements
  - Horizontally centering
    - **{justify-content: center;}**

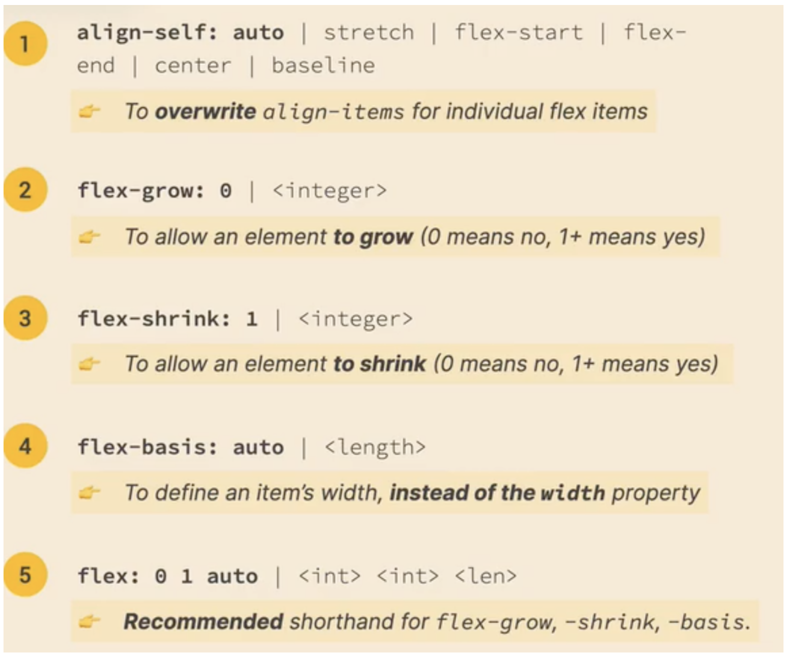


    - **{justify-content: space-between;}**



  - Characters:

    - Empty space inside a container element can be automatically divided by its child elements.
    - Align items to one another inside the parent container both vertically and horizontally.
    - vertical centering and create equal-height columns
  - Terminology
    - Flex container {display:flex;}
      - {gap:0 | <length>}: create the space between items
      - {justify-content: center | flex-start | flex-end|space-between|space-around|space-evenly;}:horizontally
      - {align-items:center | baseline | flex-end|flex-start;} :vertically
      - {flex-direction:row|row-reverse|column|column-reverse;}:define main axis
      - {flex-wrap: nowrap|wrap|wrap-reverse;}:wrap items into a new line if they are too large
      - {align-content:flex-start | flex-end|center|space-between|space-around;}:only apply for a multiple lines
    - Flex items: child elements
      - **{order:0|<integar>}**: -1 the first order, 1 means the last order, if we want to change another item to the last one, we set the order>1, such as {order:2}



1 **align-self: auto** | stretch | flex-end | center | baseline

*To **overwrite** `align-items` for individual flex items*

2 **flex-grow: 0** | <integer>

*To allow an element **to grow** (0 means no, 1+ means yes)*

3 **flex-shrink: 1** | <integer>

*To allow an element **to shrink** (0 means no, 1+ means yes)*

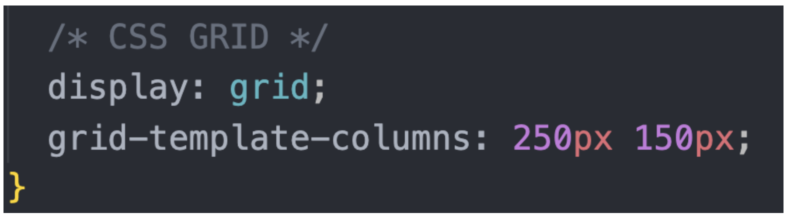
4 **flex-basis: auto** | <length>

*To define an item's width, **instead of the `width`** property*

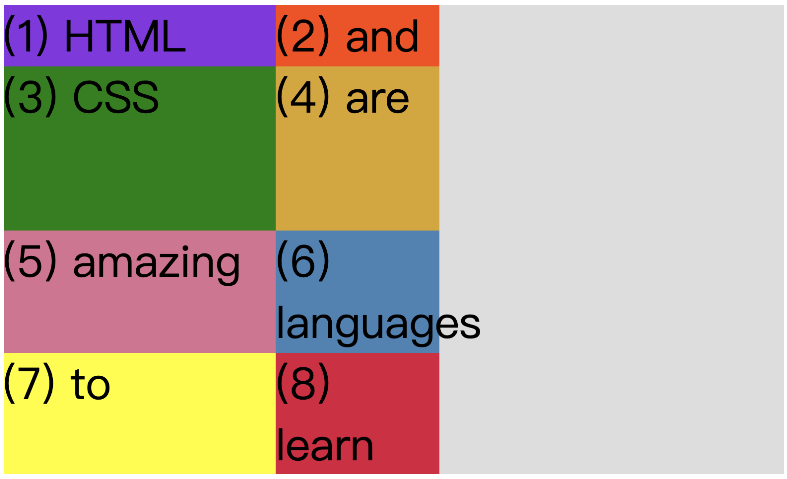
5 **flex: 0 1 auto** | <int> <int> <len>

***Recommended** shorthand for `flex-grow`, `-shrink`, `-basis`.*

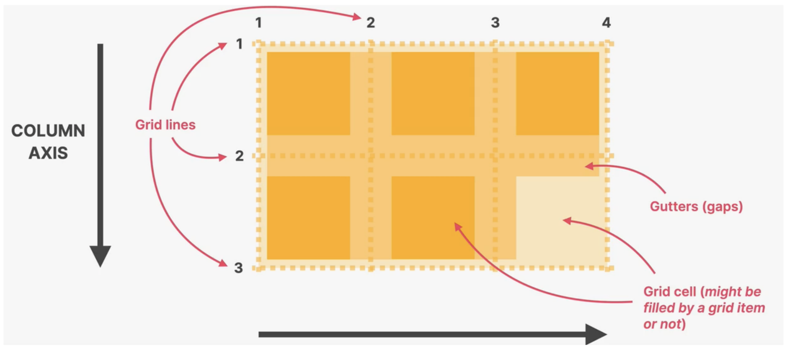
    - Main axis: the directions flex items lay out
    - Cross axis:vertical to main axis
- **CSS Grid**
  - We assign our items into two columns the first with 250px, the another is 150px. Display by **columns and rows**. The same as grid-template-rows



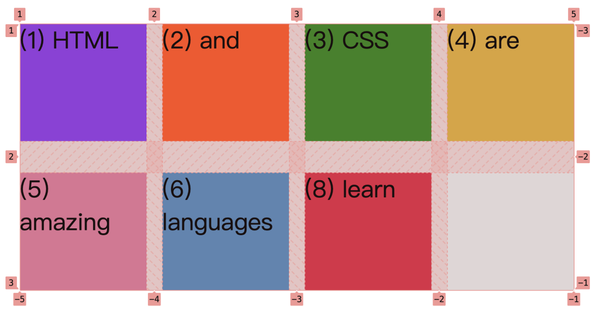
```
  /* CSS GRID */
  display: grid;
  grid-template-columns: 250px 150px;
}
```

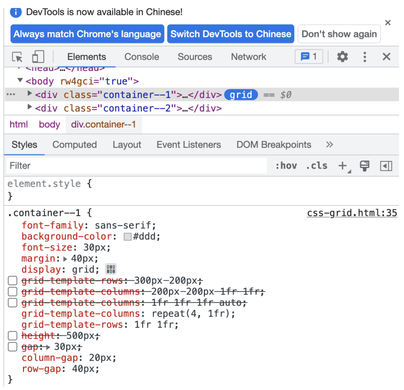


- gap/column-gap/row-gap
- Characters
  - Two-dimensional layout
  - divide a container element into columns and rows can be filled with its child elements
  - write less nested html and easier read css
  - can not replace flexbox , they can use together for 1D and 2D layout
- Terminology {diaplay:grid;}, use {display:none;} to hide css grid container
  - Grid Container
  - Grid Items
  - Gutters/Gaps: the space between two different items
  - Grid Lines
    - Grid columns: # of columns+1
    - Grid rows: # of rows +1
  - Grid Cells: Separated by grid lines, but they are not always need to be filled
  - Row Axis
  - Column Axis



- Grid track: a grid row or a grid column
- Properties
  - Grid Container
    - Establish grid row and column tracks
      - {grid-templete-rows:<size>;}
      - {grid-templete-columns:<size>;}
      - Another way to get columns separated : {grid-template-columns: **repeat**(4,1fr);}, first we state how many columns we want, then the size
    - Create Empty Space between tracks
      - {row-gap:<length>;}
      - {column-gap:<length>;}
    - Align items <u>inside</u> rows/columns(horizontally and vertically)
      - {justify-items:strech|start|center|end;}
      - {align-items:strech|start|center|end;}
    - grid inside container(if container>grid)
      - {justify-content:strech|start|center|end...;}
      - {align-content:strech|start|center|end...;}
  - Grid Items
    - place a grid item into a specific cell, based on <u>line numbers</u>
      - {grid-rows:<start line>/<end line> |span <number>;}
      - {grid-columns:<start line>/<end line> |span <number>;}
    - overwrite justify item/align items for single item
      - {justify-self:strech|start|center|end;}
      - {align-self:strech|start|center|end;}
- Placing and Spanning Grid Items



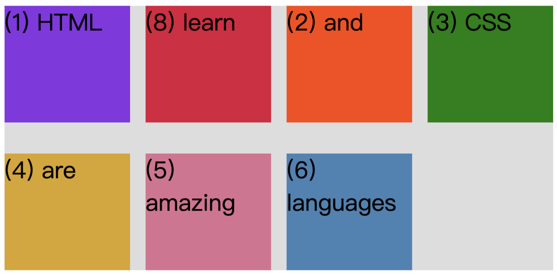


DevTools is now available in Chinese!

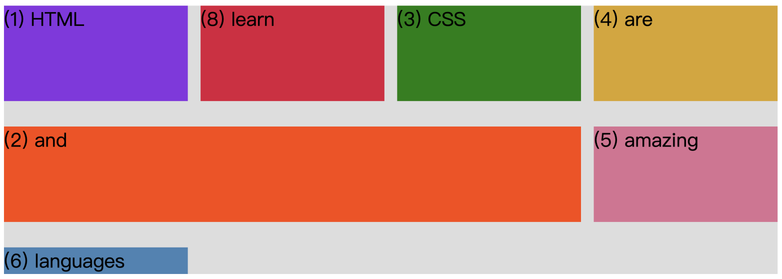
```
<head>…</head>
▼<body rw4gci="true">
 ▶<div class="container--1">…</div> grid == $0
 ▶<div class="container--2">…</div>
```

```
element.style {
}
.container--1 {                                   css-grid.html:35
   font-family: sans-serif;
   background-color: #ddd;
   margin: ▶ 40px;
   display: grid;
   ~~grid-template-rows: 300px-200px;~~
   ~~grid-template-columns: 200px-200px 1fr 1fr;~~
   ~~grid-template-columns: 1fr 1fr 1fr auto;~~
   grid-template-columns: repeat(4, 1fr);
   grid-template-rows: 1fr 1fr;
   ~~height: 500px;~~
   ~~gap: ▶ 30px;~~
   column-gap: 20px;
   row-gap: 40px;
}
```

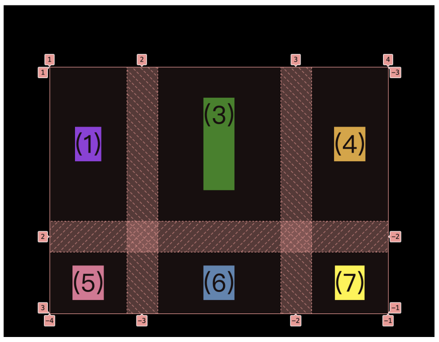
  - **.el--8{grid-column:2/3;grid-row:1/2;}** place the item 8 to the position of item2

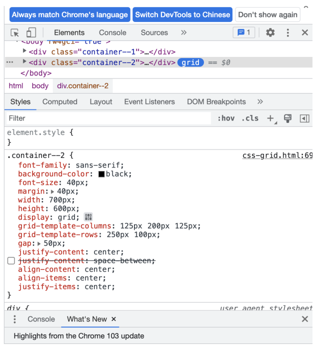


  - **.el--8{grid-column:1/4;grid-row:2;}** span cross two cells or **.el--8{grid-column:1/span3;grid-row:2;}** start from 1 span across 3 cells



- Aligning Grid Items and Tracks
  - items vertically and herizontally set center in cells





```
▸<div class="container--1">…</div>
▸<div class="container--2">…</div> grid == $0
</body>
html body div.container--2

element.style {
}
.container--2 {                                    css-grid.html:69
  font-family: sans-serif;
  background-color: ■black;
  font-size: 40px;
  margin: ▸ 40px;
  width: 700px;
  height: 600px;
  display: grid;
  grid-template-columns: 125px 200px 125px;
  grid-template-rows: 250px 100px;
  gap: ▸ 50px;
  justify-content: center;
  ~~justify-content: space-between;~~
  align-content: center;
  align-items: center;
  justify-items: center;
}
div {                                          user agent stylesheet
```

以上内容整理于 幕布文档