

Web Application Development

CSS (Cascading Style Sheets) Layout

Contents

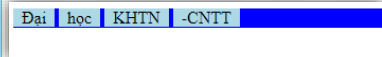
- ☐ Display
- ☐ Flexible box layout
- ☐ Grid layout
- ☐ CSS – Page Layout

Contents

- ❑ **Display**
- ❑ Flexible box layout
- ❑ Grid layout
- ❑ CSS – Page Layout

Block – Inline (default)

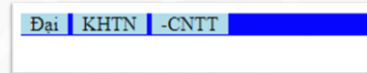
```
<html>
  <head>
    <style>
      div {
        background-color: blue;
      }
      span {
        background-color: lightblue;
        padding: 0 0.5rem;
      }
    </style>
  </head>
  <body>
    <div>
      <span>Đại</span>
      <span>học</span>
      <span>KHTN</span>
      <span>-CNTT</span>
    </div>
  </body>
</html>
```



Đại | học | KHTN | -CNTT

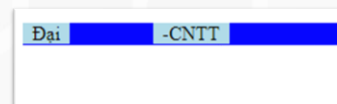
Display: none

```
<head>
  <style>
    div {
      background-color: blue;
    }
    span {
      background-color: lightblue;
      padding: 0 0.5rem;
    }
    .dnone {
      display: none;
    }
  </style>
</head>
<body>
  <div>
    <span>Đại</span>
    <span class="dnone">học</span>
    <span>KHTN</span>
    <span>-CNTT</span>
  </div>
</body>
```



Visibility: hidden

```
<head>
  <style>
    div {
      background-color: blue;
    }
    span {
      background-color: lightblue;
      padding: 0 0.5rem;
    }
    .dnone {
      display: none;
    }
    .vhidden {
      visibility: hidden;
    }
  </style>
</head>
<body>
  <div>
    <span>Đại</span>
    <span class="dnone">học</span>
    <span class="vhidden">KHTN</span>
    <span>-CNTT</span>
  </div>
</body>
```



Display: block

```
<head>
  <style>
    div {
      background-color: blue;
    }
    span {
      background-color: lightblue;
      padding: 0 0.5rem;
      display: block;
    }
  </style>
</head>
<body>
  <div>
    <span>Đại</span>
    <span>học</span>
    <span>KHTN</span>
    <span>-CNTT</span>
  </div>
</body>
```



Đại
học
KHTN
-CNTT

Display: inline-block

```
<style>
  div {
    background-color: blue;
  }
  span {
    background-color: lightblue;
    padding: 0 0.5rem;
    display: inline-block;
  }
  span:nth-child(1) {
    height: 1rem;
  }
  span:nth-child(2) {
    height: 2rem;
  }
  span:nth-child(3) {
    height: 3rem;
  }
  span:nth-child(4) {
    height: 4rem;
  }
</style>
```



Đại học KHTN -CNTT

Contents

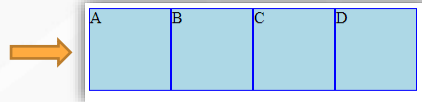
- ☐ *Display*
- ☐ **Flexible box layout**
- ☐ Grid layout
- ☐ CSS – Page Layout

Flexible box layout (flexbox)

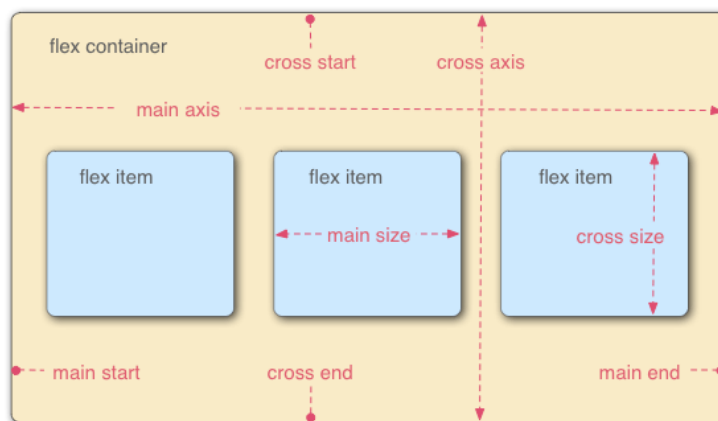
- ☐ **Flexbox** is a **one-dimensional** layout method for arranging items in rows or columns. Items flex (expand) to fill additional space or shrink to fit into smaller spaces.
- ☐ Before the Flexbox Layout module, there were four layout modes:
 - ☐ Block, for sections in a webpage
 - ☐ Inline, for text
 - ☐ Table, for two-dimensional table data
 - ☐ Positioned, for explicit position of an element
- ☐ The Flexible Box Layout Module, makes it easier to design flexible responsive layout structure without using float or positioning

Example

```
<head>
  <style>
    .container {
      display: flex;
    }
    .item {
      width: 5rem;
      height: 5rem;
      background-color: lightblue;
      border: solid 1px blue;
    }
  </style>
</head>
<body>
  <div class="container">
    <div class="item">A</div>
    <div class="item">B</div>
    <div class="item">C</div>
    <div class="item">D</div>
  </div>
</body>
```



Flex Model



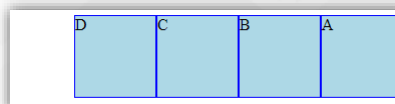
Flex Direction

❑ Flexbox provides a property called **flex-direction** that specifies which direction **the main axis** runs (which direction the flexbox children are laid out in). The following values are accepted:

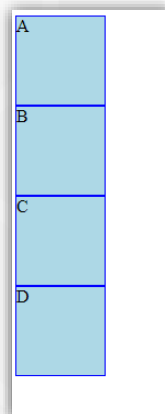
- ❑ row (default)
- ❑ row-reverse
- ❑ column
- ❑ column-reverse

Example

```
.container {  
  display: flex;  
  flex-direction: row-reverse;  
}
```



```
.container {  
  display: flex;  
  flex-direction: column;  
}
```



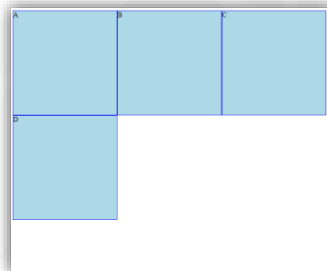
Flex Wrap

- ❑ The **flex-wrap** CSS property sets whether flex items are forced onto one line or can wrap onto multiple lines. If wrapping is allowed, it sets the direction that lines are stacked. The following values are accepted:

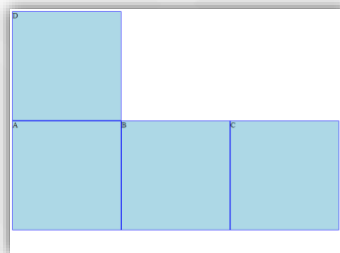
- ❑ wrap
- ❑ Nowrap (default)
- ❑ wrap-reverse

Example

```
.container {  
  display: flex;  
  flex-wrap: wrap;  
}
```



```
.container {  
  display: flex;  
  flex-wrap: wrap-reverse;  
}
```



flex-flow shorthand

- ❑ **flex-flow** is a property for setting both *flex-direction* and *flex-wrap* at the same time

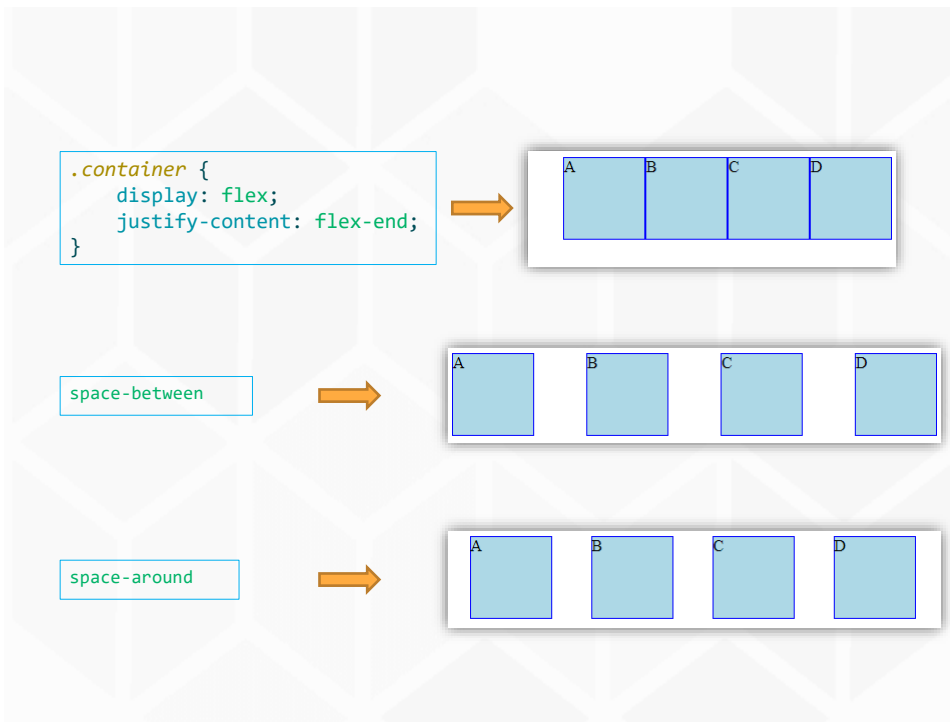
```
.container {  
  display: flex;  
  flex-flow: row wrap;  
}
```



```
.container {  
  display: flex;  
  flex-direction: row;  
  flex-wrap: wrap;  
}
```

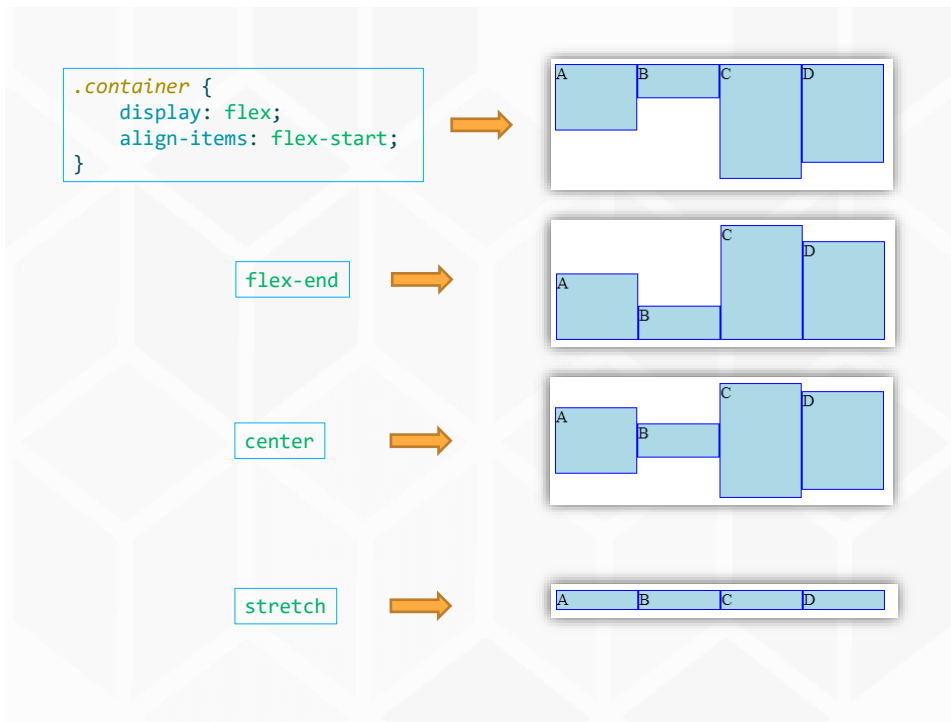
Justify Content

- ❑ The CSS **justify-content** property defines how the browser distributes space between and around content items along the **main-axis** of a flex container, and the inline axis of a grid container.
- ❑ Values:
 - ❑ flex-start
 - ❑ flex-end
 - ❑ center
 - ❑ space-between
 - ❑ space-around
 - ❑ space-evenly



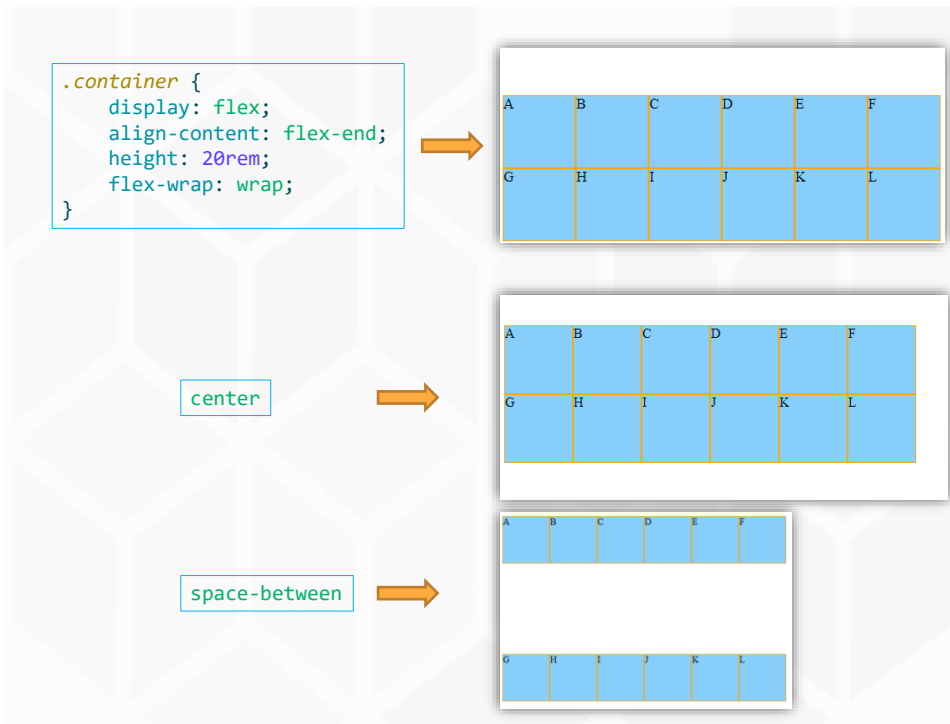
Align Items

- ❑ The CSS **align-items** property sets the **align-self** value on all direct children as a group. In **Flexbox**, it controls the alignment of items on the **Cross Axis**. In **Grid Layout**, it controls the alignment of items on the **Block Axis** within their grid area.
- ❑ Values:
 - ❑ flex-start
 - ❑ flex-end
 - ❑ center
 - ❑ stretch
 - ❑ baseline



Align Content

- ❑ The CSS **align-content** property sets the distribution of space between and around content items along a **flexbox's cross-axis** or a **grid's block axis**.
- ❑ Values:
 - ❑ flex-start
 - ❑ flex-end
 - ❑ center
 - ❑ stretch
 - ❑ space-between
 - ❑ space-around



Order

```
.a {
  order: 1;
}
.b {
  order: -2;
}
```

```
<div class="item a">A</div>
<div class="item b">B</div>
<div class="item">C</div>
<div class="item">D</div>
```



Flex Grow

```
.item {
  width: 5rem;
  height: 5rem;
  background-color: lightskyblue;
  border: solid 1px orange;
  display: flex;
  align-items: center;
  justify-content: center;
  font-size: 3rem;
  flex-grow: 1;
}
```

```
.a {
  order: 1;
  flex-grow: 2;
}
.b {
  order: -2;
  flex-grow: 0.5;
}
```



Flex Shrink

```
.item {
  width: 15rem;
  height: 5rem;
  background-color: lightskyblue;
  border: solid 1px orange;
  display: flex;
  align-items: center;
  justify-content: center;
  font-size: 3rem;
}
```

```
.a {
  order: 1;
  flex-shrink: 2;
}
.b {
  order: -2;
  flex-shrink: 0;
}
```



Flex Basis

```
.item {
  width: 10rem;
  height: 5rem;
  background-color: lightskyblue;
  border: solid 1px orange;
  display: flex;
  align-items: center;
  justify-content: center;
  font-size: 3rem;
}
```

```
.a {
  order: -1;
  flex-shrink: 2;
}
.b {
  order: 0;
  flex-basis: 20rem;
}
```



flex property

- **flex** is a property for setting *flex-grow*, *flex-shrink* and *flex-basis* at the same time

```
.flex{
  flex: 2 0 100px;
}
```

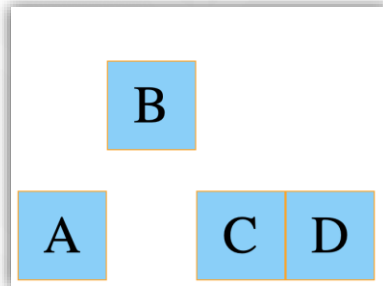


```
.flex{
  flex-basis: 100px;
  flex-grow: 2;
  flex-shrink: 0;
}
```

Align Self

```
.container {  
  display: flex;  
  align-items: flex-end;  
  height: 20rem;  
}
```

```
.b {  
  order: 0;  
  align-self: center;  
}
```



Contents

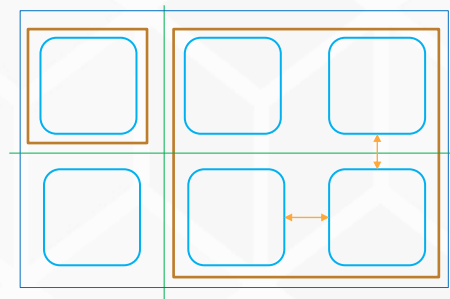
- ☐ *Display*
- ☐ *Flexible box layout*
- ☐ **Grid layout**
- ☐ CSS – Page Layout

Grid layout

- ❑ **CSS grid layout** is a **two-dimensional** layout system for the web. It lets you organize content into rows and columns and offers many features to simplify the creation of complex layouts.
- ❑ A **grid** is a collection of horizontal and vertical lines creating a pattern against which we can line up our design elements. They help us to create layouts in which our elements won't jump around or change width as we move from page to page, providing greater consistency on our websites.

CSS Grid Layout Terminology

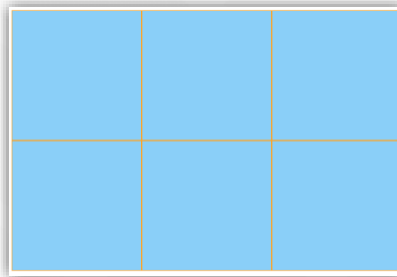
- ❑ **Grid Container**
- ❑ **Grid Lines**
- ❑ **Grid Cell**
- ❑ **Grid Area**
- ❑ **Grid Row**
- ❑ **Grid Column**
- ❑ **Grid Gap**



Defining a grid

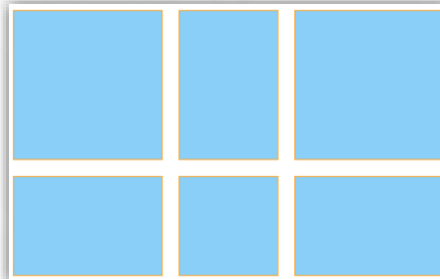
```
.container {  
  display: grid;  
  grid-template-columns: 150px 150px 150px;  
  grid-template-rows: 150px 150px;  
}  
.cell {  
  background-color: lightskyblue;  
  border: solid 1px orange;  
}
```

```
<div class="container">  
  <div class="cell"></div>  
  <div class="cell"></div>  
  <div class="cell"></div>  
  <div class="cell"></div>  
  <div class="cell"></div>  
  <div class="cell"></div>  
</div>
```



Gaps between tracks

```
.container {  
  display: grid;  
  grid-template-columns: 150px 100px 150px;  
  grid-template-rows: 150px 100px;  
  grid-gap: 1rem;  
  gap: 1rem;  
}
```



FR (Fraction) Unit

```
.container {  
  display: grid;  
  grid-template-columns: 150px 1.5fr 1fr;  
  grid-template-rows: 150px 100px;  
  grid-gap: 0.25rem;  
}
```



Grid Start - End

```
.s {  
  grid-row: 1 / 3;  
  grid-column: 2 / 4;  
}
```

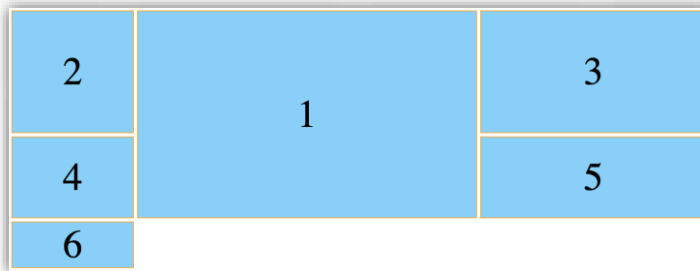
```
.s {  
  grid-row: 1 / span 2;  
  grid-column: 2 / span 2;  
}
```



Named Grid Lines

```
.container {
  display: grid;
  grid-template-columns:
    [column-1] 150px
    [column-2] 1.5fr
    [column-3] 1fr;
  grid-template-rows: 150px 100px;
  grid-gap: 0.25rem;
}
```

```
.s {
  grid-row: 1 / 3;
  grid-column: column-2 / column-3;
}
```



Template Areas

```
.container {
  display: grid;
  grid-gap: 0.5rem;
  grid-template-areas:
    "top top"
    "middle-1 middle-2 "
    "bottom bottom";
}
```

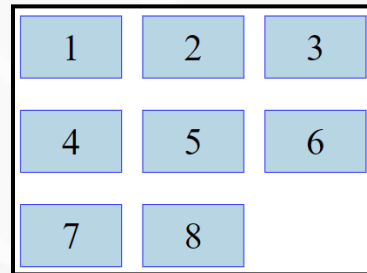
```
.top {
  grid-area: top;
}
.middle-1 {
  grid-area: middle-1;
}
.middle-2 {
  grid-area: middle-2;
}
.bottom {
  grid-area: bottom;
}
```



Implicit Grids

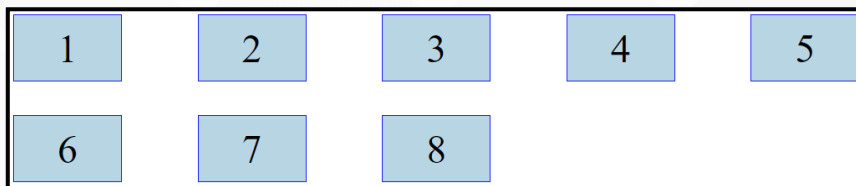
- ❑ **Implicit grid** extends the defined **explicit grid** when content is placed outside of that grid, such as into the rows by drawing additional grid lines

```
.container {
  display: grid;
  grid-template-columns: repeat(3, 1fr);
  grid-auto-rows: 100px;
  gap: 20px;
}
```



Implicit Grids (cont.)

```
.container {
  display: grid;
  grid-template-columns: repeat(auto-fit, minmax(200px, 1fr));
  grid-auto-rows: minmax(100px, auto);
  gap: 20px;
}
```



Contents

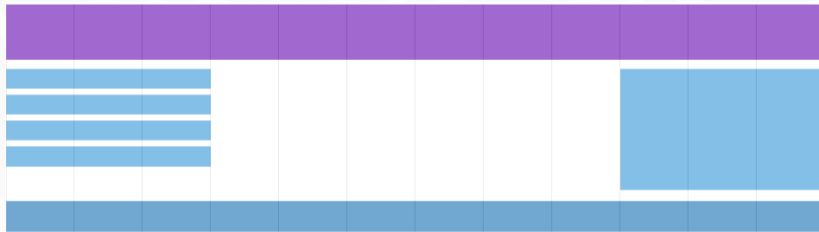
- ☐ *Display*
- ☐ *Flexible box layout*
- ☐ *Grid layout*
- ☐ **CSS – Page Layout**

CSS - Page Layout

- ☐ **CSS page layout** techniques allow us to take elements contained in a web page and control where they're positioned relative to the following factors: their default position in normal layout flow, the other elements around them, their parent container, and the main viewport/window.

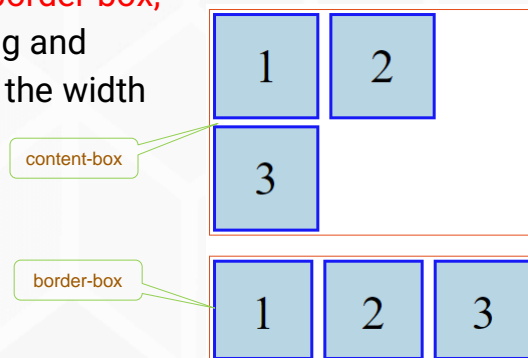
Grid-View

- ❑ Many web pages are based on a grid-view, which means that the page is divided into columns. Using a grid-view is very helpful when designing web pages. It makes it easier to place elements on the page.
- ❑ A responsive grid-view often has 12 columns, and has a total width of 100%, and will shrink and expand as you resize the browser window.



Box-sizing

- ❑ The CSS **box-sizing** property allows us to include the padding and border in an element's total width and height.
- ❑ If you set **box-sizing: border-box;** on an element, padding and border are included in the width and height.



Responsive Web Design

- ☐ Responsive web design makes your web page look good on all devices.
- ☐ Responsive web design uses only HTML and CSS.
- ☐ Responsive web design is not a program or a JavaScript.



Strategies

1. Fluid & Flexible Layouts

- ☐ Elements can automatically resize, hide, shrink, or enlarge

2. Flexible Images

- ☐ Images can scale nicely to fit any browser size

3. CSS3 Media Queries

- ☐ Define different styles for different browser sizes

Fluid & Flexible Layout & Image

- ❑ Setting the view port to give the browser instructions on how to control the page's dimensions and scaling

```
<meta name="viewport" content="width=device-width, initial-scale=1.0">
```

- ❑ Set the width property to 100%, the image will be responsive and scale up and down

```

```

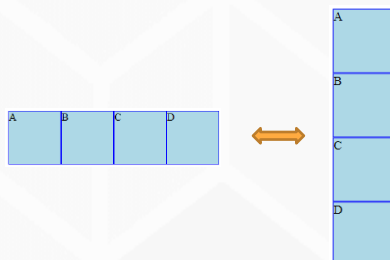
CSS3 Queries

- ❑ Use media attribute

```
<link rel="stylesheet" media="screen" href="style.css" />
```

```
<link rel="stylesheet" media="only screen and (min-width:320px) and (max-width:568px)" href="mobile.css" />
```

```
<link rel="stylesheet" media="only screen and (min-width:768px) and (max-width:980px)" href="tablet.css" />
```

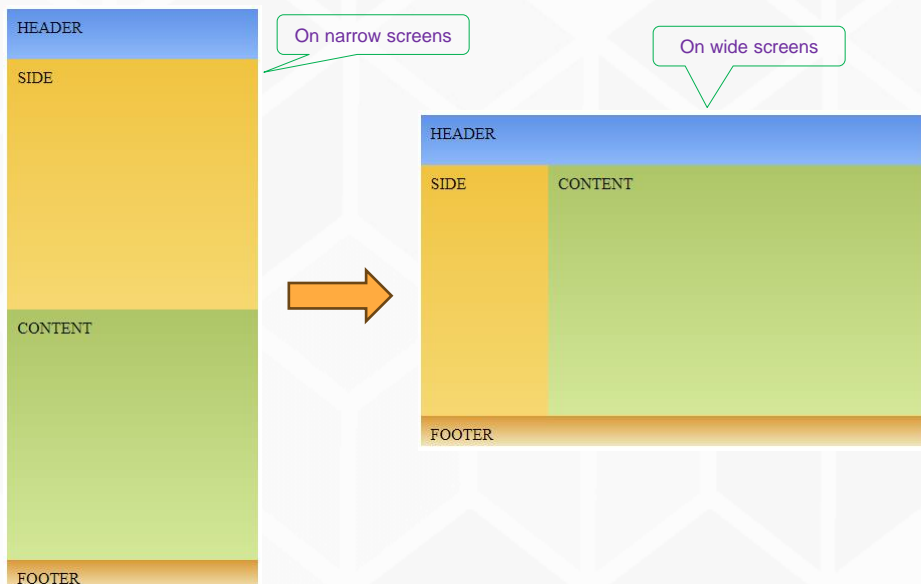


CSS3 Queries (cont.)

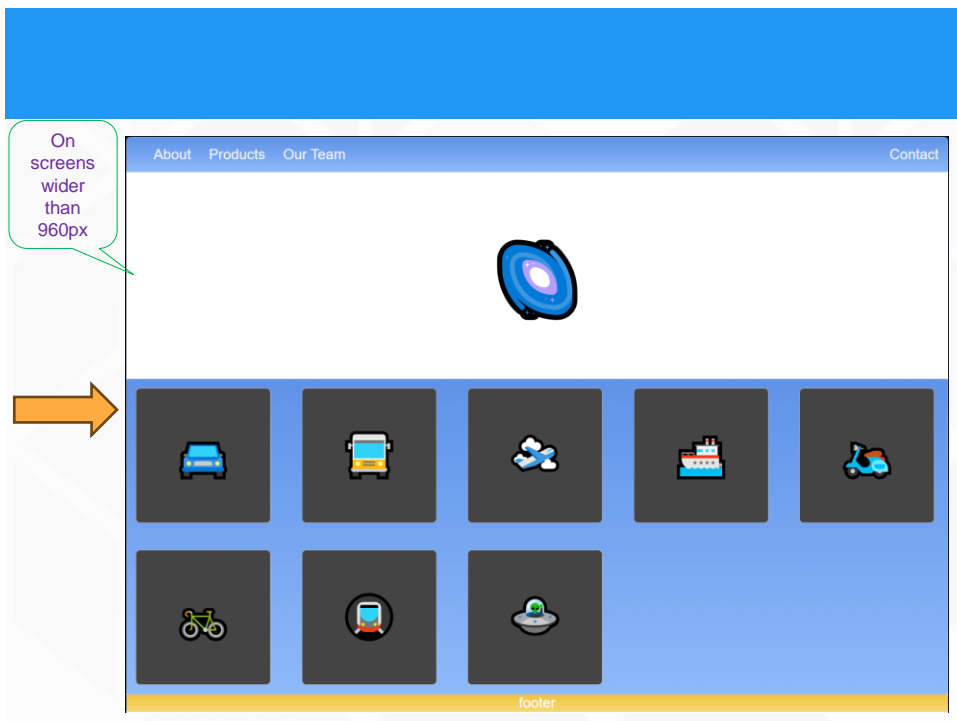
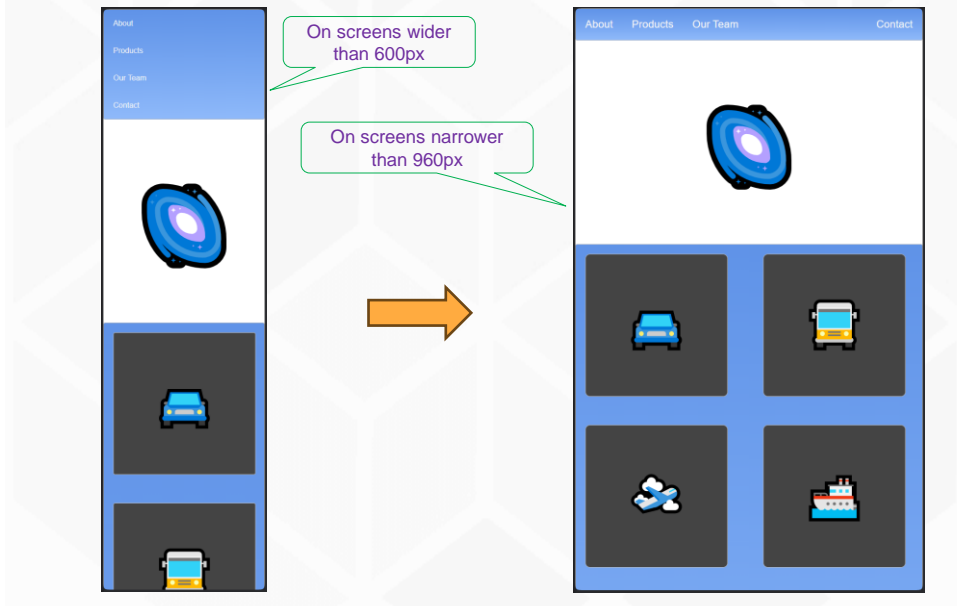
❑ Use media query

```
@media (max-width:600px) {  
  /* Styles go here */  
}  
  
@media (min-width:700px) {  
  /* Styles go here */  
}  
  
@media only screen and (min-width:320px) and (max-width: 568px) {  
  /* Styles go here */  
}  
  
@media only screen and (min-width:768px) and (max-width: 980px) {  
  /* Styles go here */  
}
```

Example 1



Example 2



Contents

- ☐ *Display*
- ☐ *Flexible box layout*
- ☐ *Grid layout*
- ☐ *CSS – Page Layout*
- ☐ **Exercises**

Exercises



- ☐ Complete Example 1 with **Float** layout technique and make it **responsive**.
- ☐ Complete Example 1 with **Flexbox** layout technique and make it **responsive**.
- ☐ Complete Example 1 with **Grid** layout technique and make it **responsive**
- ☐ Complete Example 2 by combining layout techniques and making it **responsive**