

# Web page layout

## Chapter 4

# Outline

2

- A. Normal flow
- B. Floating
- C. Column and grid layout
- D. Flexible box model

# CSS Box

3

- Every element is displayed in one (or more) box
  - ▣ We say that this element *generates* the box
- The box has content, padding, border and margin.

<body>



<h1>Structure of HTML</h1>



<p>An HTML doc has a



<em>tree structure</em>,

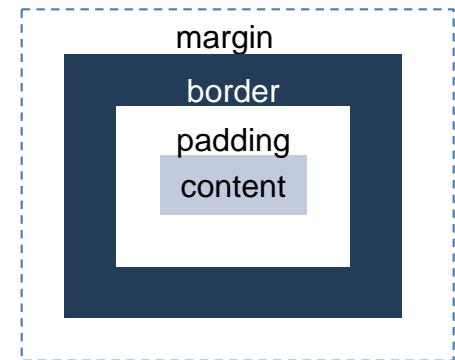


where a <span>parent</span> element  
contains some <span>children</span>  
elements.



</p>

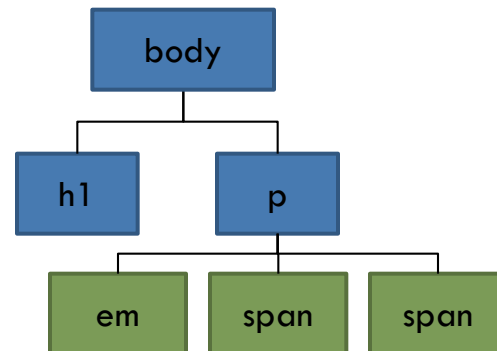
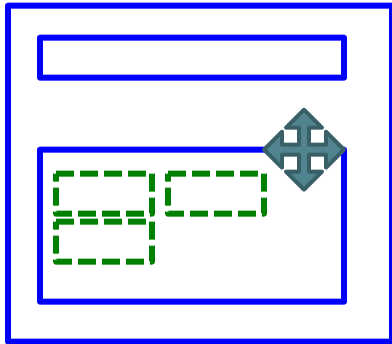
</body>



# Displaying elements in document tree

4

- Typically, an element is rendered inside the box generated by an ancestor.
  - ▣ Exceptions: overflow and absolute positioning
- The exact position is determined by its positioning scheme.
  - ▣ You can fine-tune the position with some properties.



# Positioning schemes

5

- CSS defines several positioning schemes
  - ▣ Normal flow (CSS1)
  - ▣ Floating (CSS1)
  - ▣ Absolute positioning (CSS2)
  - ▣ Flexible box (CSS3)
- We specify a positioning scheme for a box using two properties **display**, **position**, and **float**.
  - ▣ The default values **position: static** and **float: none** select normal flow
  - ▣ **position: absolute / relative / fixed** selects abs pos

# Part A. Normal flow

6

- Normal flow is the default positioning scheme
  - ▣ `position: static; float: none;`
  - ▣ Boxes arranged by normal flow are sometimes known as **static boxes**
- Normal flow arranges elements according to source order and box types
  - ▣ **Inline boxes** follow the flow of a line. They are laid out from left to right, line by line.
  - ▣ **Block boxes** are laid out vertically, from top to bottom.

# Inline box vs. Block box

7

Normal flow lays out inline and block boxes in different way.

- **Inline box**, generated by an element with **display: inline**
  - ▣ By default, inline elements (e.g. `<a>` `<img>` `<span>` `<input>` `<button>`) generate inline boxes
- **Block box**, generated by an element with **display: block**
  - ▣ By default, block elements (e.g. `<h1>` `<p>` `<div>` `<ul>`) generate block boxes
- You can change the box type of any elements
  - ▣ E.g. `img { display: block; }` lays out images in its own block

# Other box types

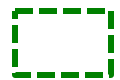
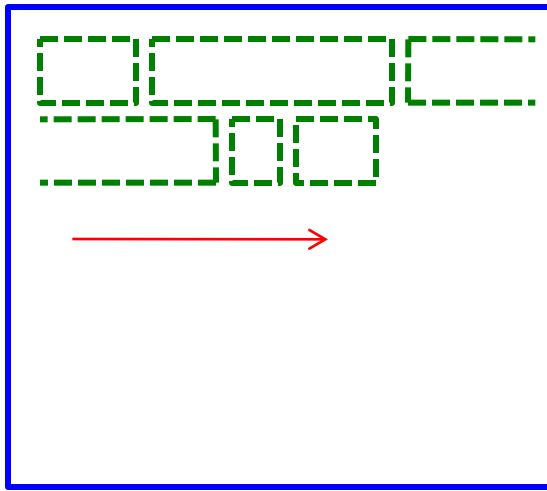
8

- CSS defines other box types
  - ▣ `display: none` – no box is generated. Normal flow ignores the element in layout
  - ▣ `display: list-item` – similar to block, but insert a bullet
  - ▣ `display: table`, and others – special box types for table, rows, columns, cells, etc
  - ▣ `display: flex` – flexible box model in CSS3
- `visibility: hidden / visible` only hides or shows an element. Normal flow still allocates space for the element.

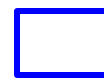
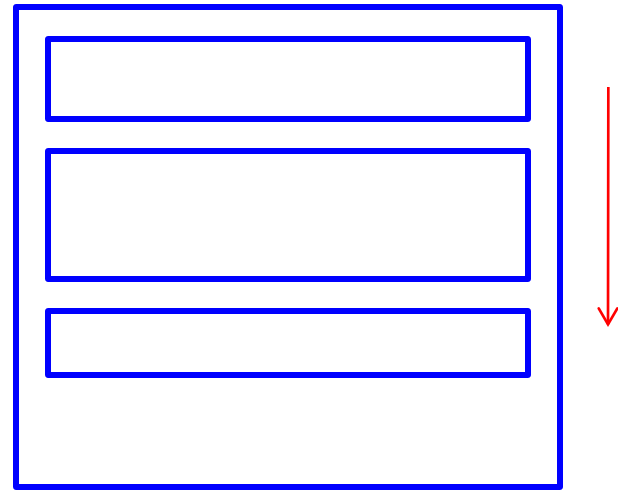


# Boxes in Normal Flow

9



Inline boxes laid out  
inside a block box



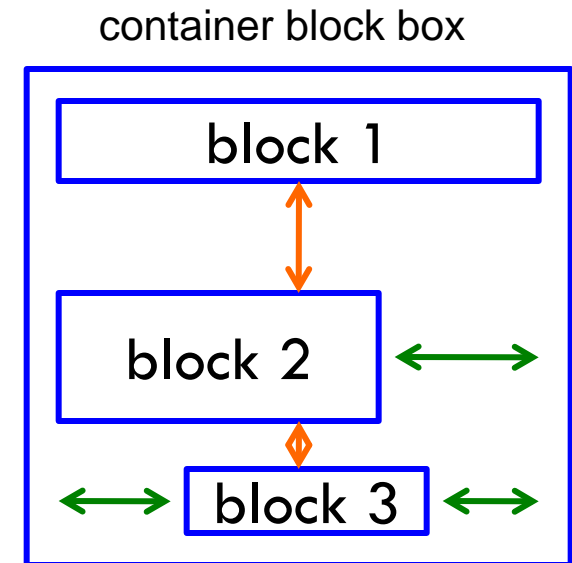
Block boxes laid out  
inside a block box

*Elements flows horizontally / vertically inside the nearest ancestor element that generates a block box.*

# Normal flow of block boxes

10

- Block boxes are arranged vertically inside the **container block**, which is the block box generated by their parent element.
  - ▣ Fixed or relative width and height
  - ▣ Left and right margins adjust the horizontal position
  - ▣ Top and bottom margins adjust the vertical position



 Vertical margins

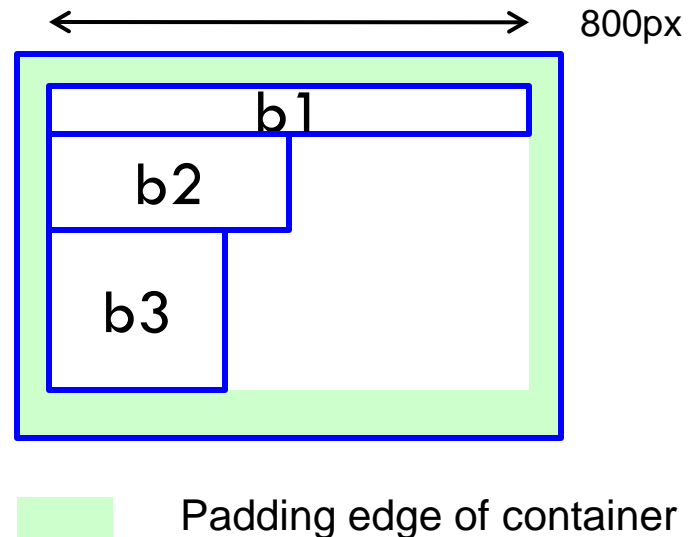
 Horizontal margins

 Box border

# Width of block box

11

```
#container { width: 800px;  
  padding: 30px; }  
#b1 { width: auto; }  
#b2 { width: 400px; }  
#b3 { width: 30%; }
```

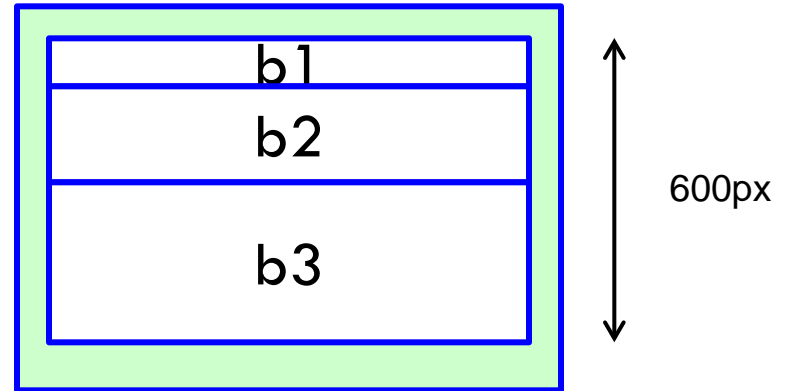


- Default **width: auto** stretches the width of the box to fill the container.
- Fixed value, e.g. width: 400px
- Percentage, e.g. width: 30%, calculated with respect to the width of the content area of the container

# Height of block box

12

```
#container { height: 600px;  
  padding: 30px; }  
#b1 { height: auto; }  
#b2 { height: 200px; }  
#b3 { height: 50%; }
```



 Padding edge of container

- Default **height: auto** makes the height large enough to contain the content of the box
- Fixed value, e.g. height: 200px
- Percentage, e.g. height: 50%, calculated with respect to the height of the content area of the container. (This is applicable only when the container height does not depend on its content.)

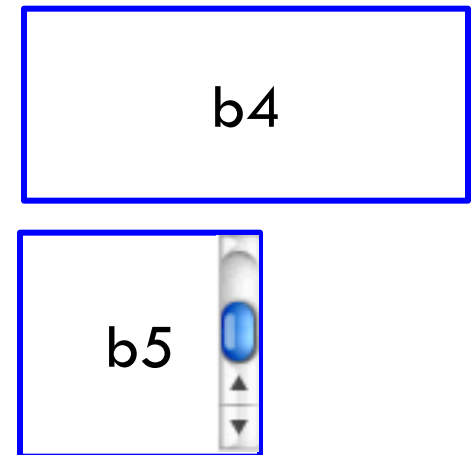
# The overflow property

13

- If the box has fixed **height**, it may be too small to contain all its content. **overflow** specifies what should happen
  - ▣ **overflow: visible** (default) - overflow spills over the box.
  - ▣ **overflow: hidden** - overflow can not be seen.
  - ▣ **overflow: scroll** - the box scrolls to accommodate the overflow. Always show scroll bars.
  - ▣ **overflow: auto** - similar to scroll, but only show scroll bars when necessary.

```
#b4 { width: 800px; }  
  
#b5 { width: 400px;  
      height: 200px;  
      overflow: auto; }
```

This example assumes that #b4 and #b5 have the same amount of text content.



# Constrain the dimension

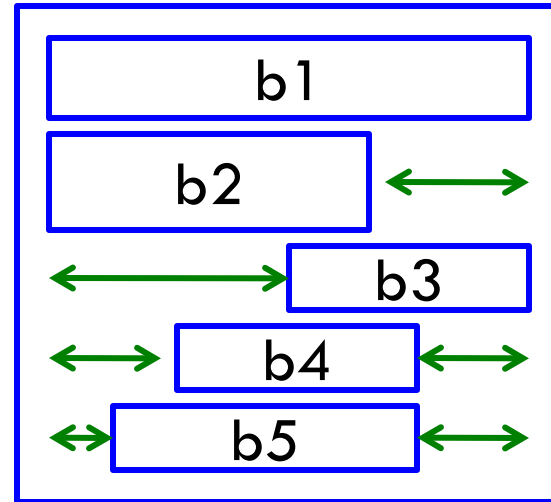
14

- ❑ A box with `width: auto` or a `percentage` width changes its width when the browser window resizes
- ❑ Low readability when the box is too narrow or too wide
- ❑ The properties `min-width` and `max-width` set limit for the width
- ❑ The properties `min-height` and `max-height` set limits for the height

# Left and Right Margins

15

```
#container { width: 800px; }  
#b1 { width: auto }  
#b2 { width: 500px;  
      margin-right: auto; }  
#b3 { width: 400px;  
      margin-left: auto; }  
#b4 { width: 400px;  
      margin-left: auto;  
      margin-right: auto; }  
#b5 { margin-left: 100px;  
      margin-right: 200px; }
```




Container padding and vertical margins are added in this diagram for clarity.


By default, `margin-left=0` and `margin-right=0`. If one of `width`, `margin-left` and `margin-right` is `auto`, it will extend to fill the container. If both margins are `auto`, the box will be centered horizontally.

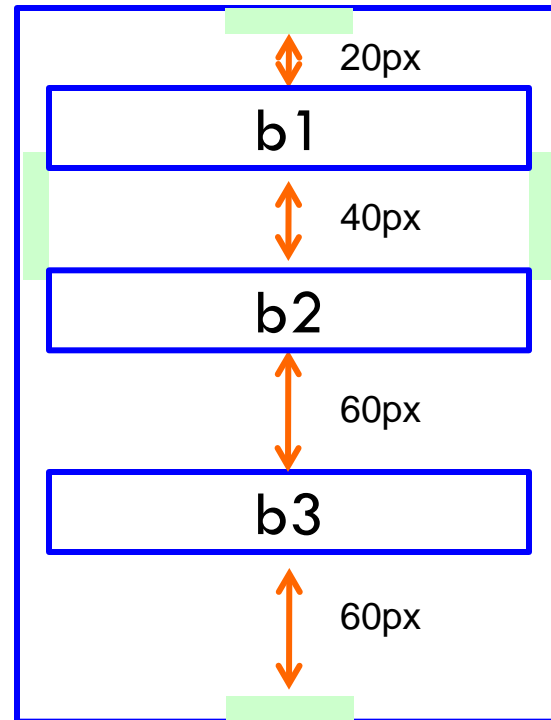
# Top and Bottom Margins

16

```
#container { padding: 10px; }  
#b1 { margin: 20px 0; }  
#b2 { margin: 40px 0; }  
#b3 { margin: 60px 0; }
```

 Padding of container

 Vertical margins



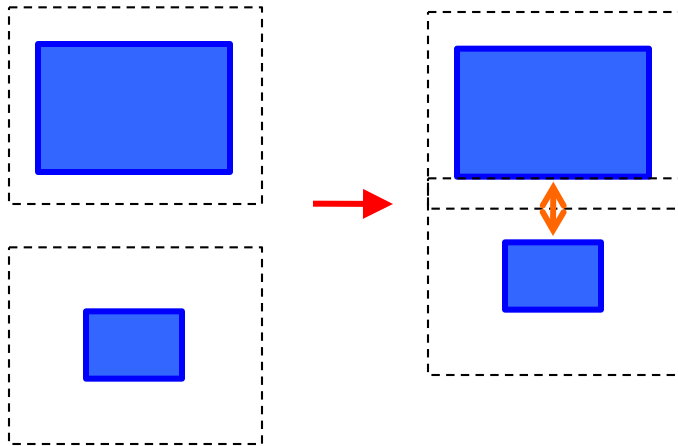
Vertical spacing between two adjacent boxes is the maximum of the bottom margin of the upper box and the top margin of the lower box.



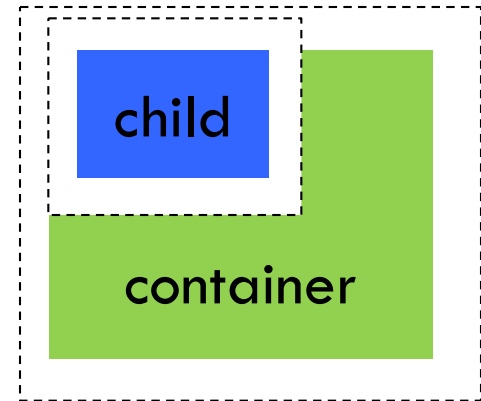
# Margin Collapsing

17

When two vertical margins touch, they collapse.



*The bottom margin of an upper box and the top margin of a lower box collapse to the maximum of the two margins.*



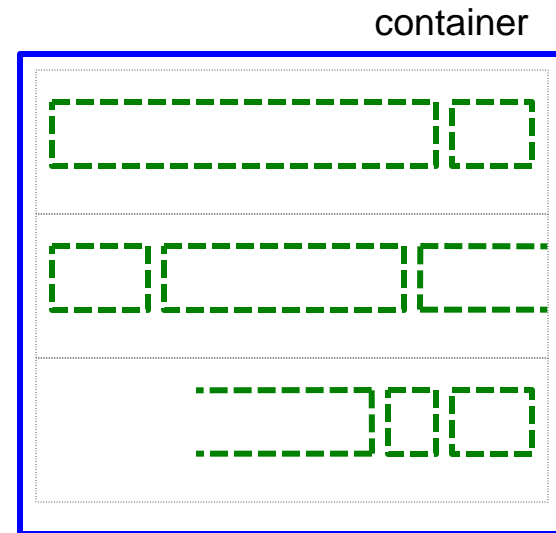
*When a container block box has no border and padding, the top margin of a child box will touch its top margin. These two top margins will collapse also.*

<http://reference.sitepoint.com/css/collapsingmargins>

# Normal flow of inline boxes

18

```
#container {  
  display: block;  
  line-height: 2;  
  text-align: right;  
}
```

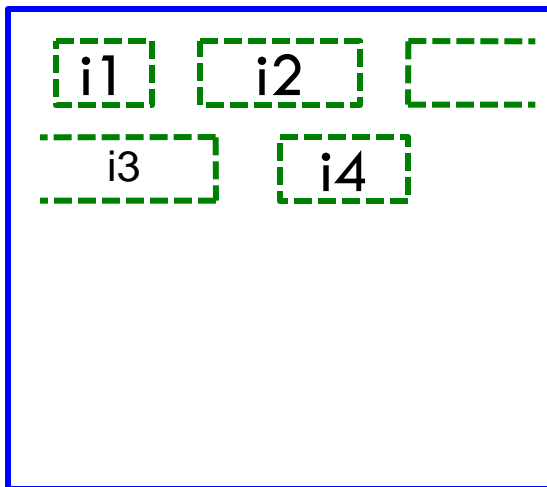


Inline boxes are laid out on line boxes stacked from top to bottom inside a block box. The property `line-height` gives the minimum height of the line boxes. Each line box should be tall enough to keep the content of inline boxes on the line.

The property `text-align` of the container also affects the positions of inline boxes in the line boxes.

# Padding and Margins

19



```
#i1, #i2, #i3, #i4 {  
  display: inline;  
  margin-left: 10px;  
  margin-right: 10px;  
}
```

The horizontal padding and margins are observed when laying out inline boxes in lines. The vertical padding and vertical margins *do not* affect line height.

# Width and height of inline boxes

20

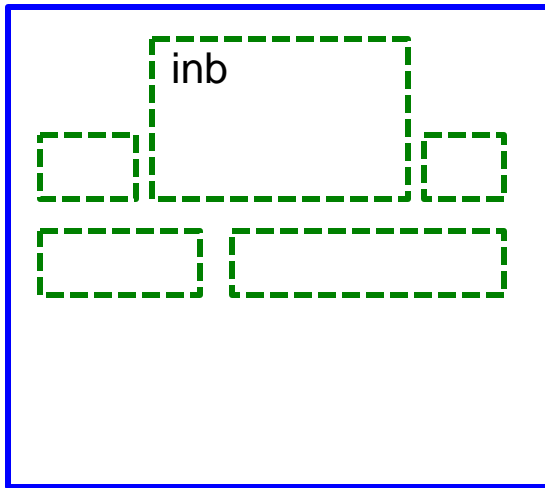
```
<style>
  p#xy span { width: 100px; height: 2em; }
  p#xy img { width: 60px; height: 60px; }
</style>

<p id='xy'>
  Two inline boxes: a span <span>like this</span>
  and an image <img src='tick.png' alt=''/>.</p>
```

You can set the width and height of 'replaced elements' like `<img>` `<iframe>` and `<object>`. On the other hand, the browser ignores the width and height setting for other ('non-replaced') inline boxes. The browser fits the box size according to the content.

# Inline-block box

21



```
#inb {  
  display: inline-block;  
  width: 100px;  
  height: 50px; }
```

A special kind of inline box, specified with `display: inline-block`, behaves as if a block box is embedded inside an inline box. You can set the dimension of such inline-block box, and the line box will be tall enough to contain its margin and padding, not just its content.

# vertical-align for text

22

**vertical-align** changes the vertical position of inline boxes in line box. The following values are common for text.

By default, text of **different size** and *different font* are aligned on baseline.

In addition, you can also align them as superscript <sup>like this</sup> and subscript <sub>like this</sub>

value	meaning
baseline	(default) align this element baseline with the baseline of the default font of the line
super	Align this element as a superscript of the default font of the line
sub	Align this element as a subscript of the default font of the line

# vertical-align for images

23

The following values are commonly used with image

value	meaning
bottom	Align the bottom of this element with the lowest element on this line
middle	Align this element in the middle of this line
top	Align the top of this element with the top of the highest element on this line

Here are several smileys  of different sizes 

# Further reading

24

- vertical-align
  - ▣ <http://css-tricks.com/what-is-vertical-align/>
- Negative margin moves a box or its neighbors
  - ▣ <http://www.smashingmagazine.com/2009/07/27/the-definitive-guide-to-using-negative-margins/>
- Vertical alignment. Vertical centering
  - ▣ <http://phrogz.net/css/vertical-align/index.html>



# Laboratory

25

- Lab 4-1: normal flow of block boxes
  - ▣ Left / right margins to adjust horizontal position
  - ▣ Top / bottom margins to adjust vertical position
    - Margin collapsing
  - ▣ Percentage height
- Lab 4-2: normal flow of inline boxes
  - ▣ Box properties that normal flow respects in arranging inline content in line boxes

# Part B. Floating

26

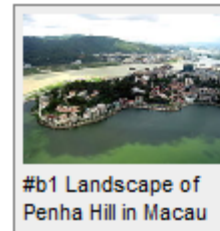
- A float is a box that is shifted to the left or right of its container box
  - ▣ Selected by `float: left` or `float: right`
  - ▣ Inline content flows along a floated box
  - ▣ Floated boxes are laid out side by side

This diagram illustrates how text flows around a float.



The image is here (X)  
After the image, here  
are some more  
sentence. Notice how  
the image is taken away  
from the normal flow of  
inline boxes, and how  
the line boxes of content  
alongside a float are

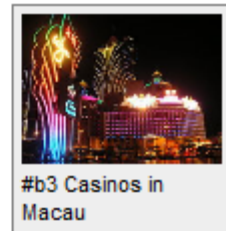
shortened to make room for the float box (including margins).



#b1 Landscape of  
Penha Hill in Macau



#b2 Macau viewed  
from Macau Museum



#b3 Casinos in  
Macau

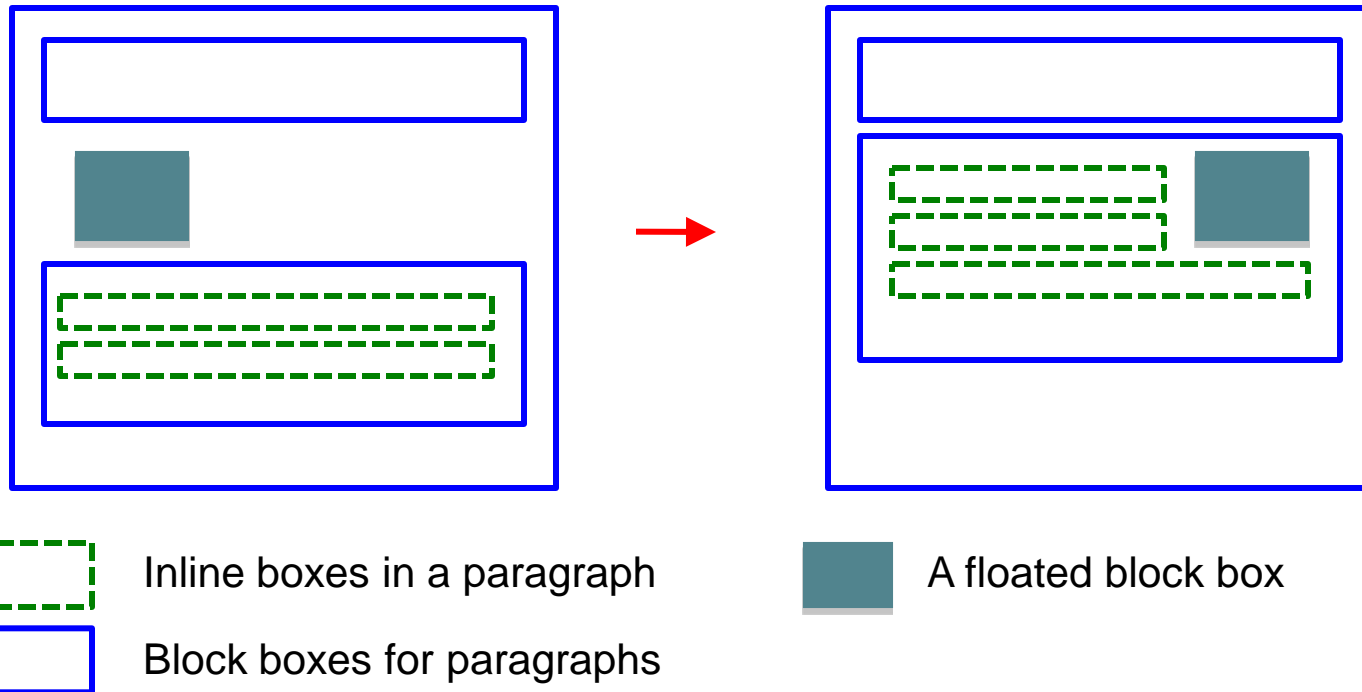
# Basic operation of floating

27

- Determine the ‘static position’ of the float, i.e. its position when laid out in normal flow
- Shift the float to left / right of the container box
- Remove the float from the flow
  - ▣ Elements after a float move up, as if the float does not exist
  - ▣ However, line boxes created after the float are shortened to make room for it
  - ▣ Do not affect preceding blocks

# Floating and Normal Flow

28



*Block boxes are positioned as if the float does not exist, but inline boxes flow around the float.*

# More about floats

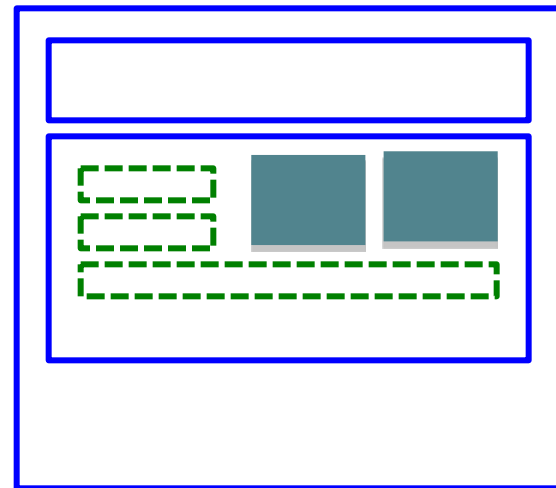
29

- You can float both inline and block boxes
  - When an inline box floats, it changes to block box
  - You should specify width of floats
- No margin collapsing on floats
  - between two floats
  - between a float and a static box

# Multiple floats

30

- When adjacent elements in HTML source code are floated to the same side, they are arranged side-by-side
- ▣ If no enough space, a float goes to the next line below the previous float



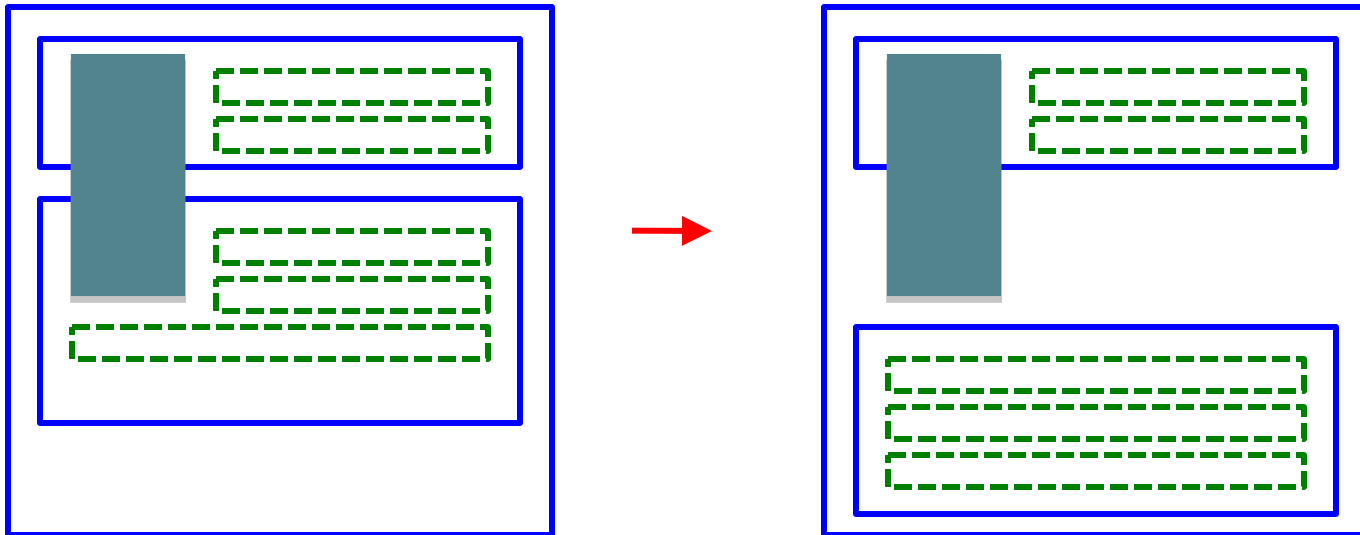
# Clearing from floats

31

- ❑ Clearing moves a block box down until it does not overlap floats
- ❑ The `clear` property
  - ❑ `clear: left` – clear on the left side
  - ❑ `clear: right` – clear on the right side
  - ❑ `clear: both` – clear on both side
  - ❑ `clear: none` (default)

# Clearing from floats

32



A float can affect line boxes of more than one block box. In this example, we use `clear: left` to clear the second paragraph from any floats on the left.



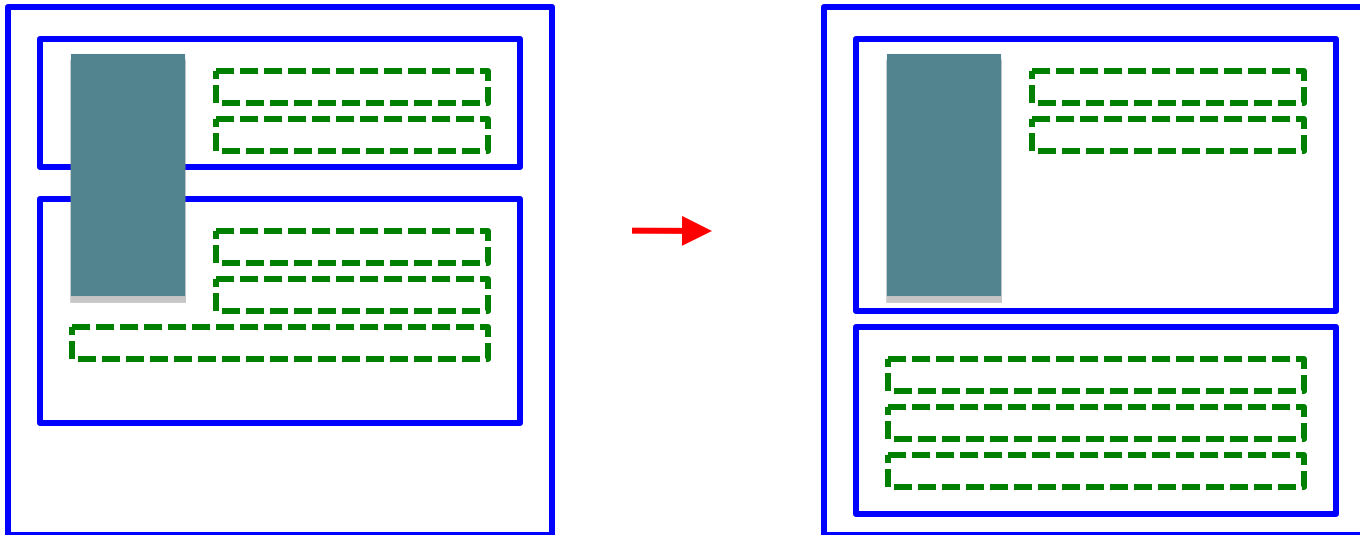
# Clear the container

33

- The height of a container box with `height: auto` is only large enough to contain normal flow content inside it
  - ▣ Floats may overflow
- Two common methods to clear the container
  - ▣ Set `overflow: auto` for the container
  - ▣ `Float the container` (note: you can float a box inside another float)
  - ▣ Ref. <http://blogs.sitepoint.com/2005/02/26/simple-clearing-of-floats/>

# Clearing the container

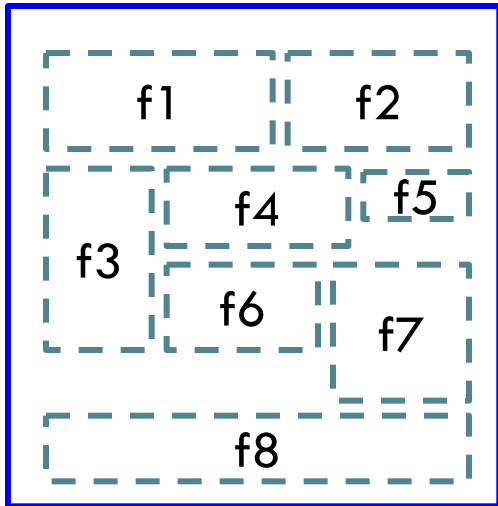
34



Clearing the container prevents a descendant float to affect other boxes.

# Floating multiple boxes

35



Floating box



Container block box

```
div.container div { float: left; }
```

```
<div class="container">  
  <div id="f1"> ... </div>  
  <div id="f2"> ... </div>  
  ...  
  <div id="f8"> ... </div>  
</div>
```

When there is enough room, a float is put beside a previous one (e.g. f2). Otherwise, it may go to the beginning of the next row (e.g. f3) or under a previous one (e.g. f6).

# Laboratory

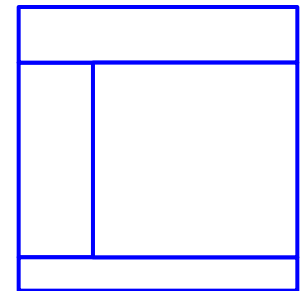
36

- Lab 4-3. Basics of floating
  - ▣ Floating inline content
  - ▣ Floating block boxes. Clearing.
- Lab 4-4. Floating for horizontal layout

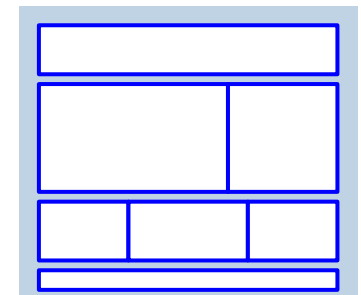
# Part C. Web page layout

37

- Modern web pages usually divide their content into boxes and arrange them in rows and columns.
- HTML markup for boxes
  - ▣ `<div>`
  - ▣ HTML5 sectioning elements
- Implementing layout
  - ▣ Floating
  - ▣ Fixed width vs. liquid width



Column layout

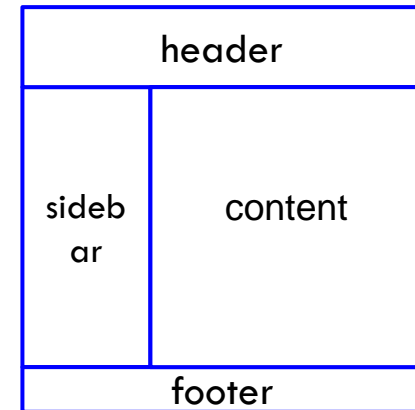


Grid layout

# Using <div> or sectioning elements

38

- It is common to use <div> for the boxes in page layout
  - ▣ Describe the function of a box with the **id** or **class** attribute
- You can also use HTML5 sectioning elements if appropriate.



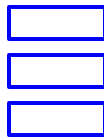
```
<body>
  <div id='header'>..
```

```
<body>
  <header>..
```

# Basic strategy in CSS layout

39

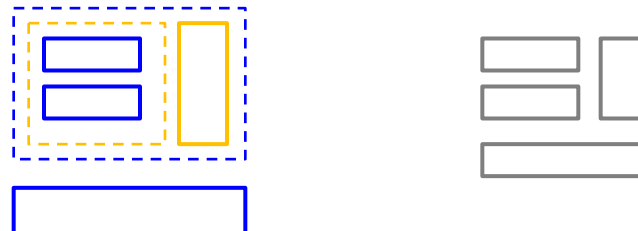
- Normal flow arranges block boxes vertically



- Floating arranges block boxes horizontally in a row.



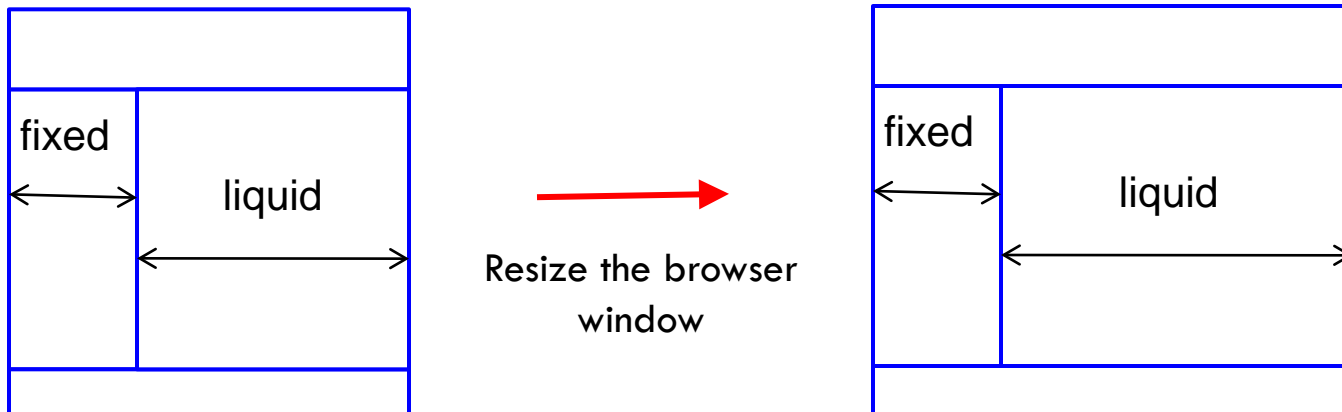
- You can nest `<div>` appropriately to mix the layout direction. Usually, you need to add `<div>` to group boxes.



# Liquid vs. fixed width

40

- A box with liquid width fills the remainder of the browser window
  - ▣ Liquid width also known as flexible, fluid or elastic.

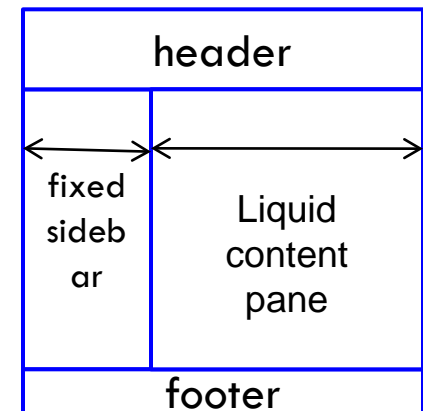




# Column layout

41

- We will use CSS to make a two-column layout with the following boxes
  - **header** contains logo
  - A sidebar **div#sidebar** has fixed width
  - Content pane **div#content** has liquid width. It is usually taller than the sidebar
  - **footer** contains copyright info

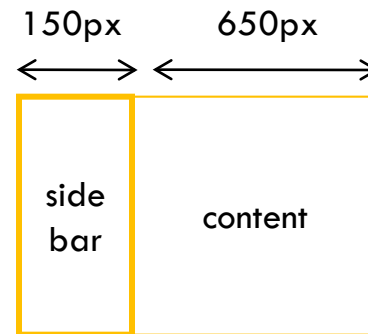


# Float both columns

42

- One way to arrange the columns in a row is to float all columns
- Shortcoming: you must fix the width of both columns. Non-flexible layout

```
#sidebar {  
  float: left;  
  width: 150px;  
}  
#content {  
  float: left;  
  width: 650px;  
}
```



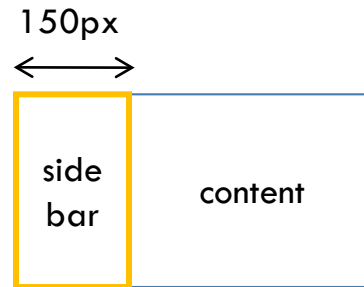
```
<body>  
  <div id="sidebar">..  
  <div id="content">..  
</body>
```

# Float one column

43

- Another method is to float the sidebar only
- The content pane uses the default width: auto.
- Benefit: liquid width
- To prevent content of the content pane to wrap around the sidebar , add a left margin to the content pane.

```
#sidebar {  
  float: left;  
  width: 150px;  
}  
#content {  
  margin-left: 150px;  
}
```



```
<body>  
  <div id="sidebar">..  
  <div id="content">..  
</body>
```

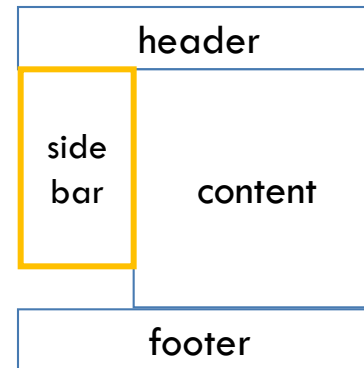
# Two columns with header & footer

44

- Header, content and footer follow normal flow
- The floated sidebar is under the header because it floats to the left from its static position

```
#sidebar { float: left; width: 150px; }  
#content { margin-left: 150px; }
```

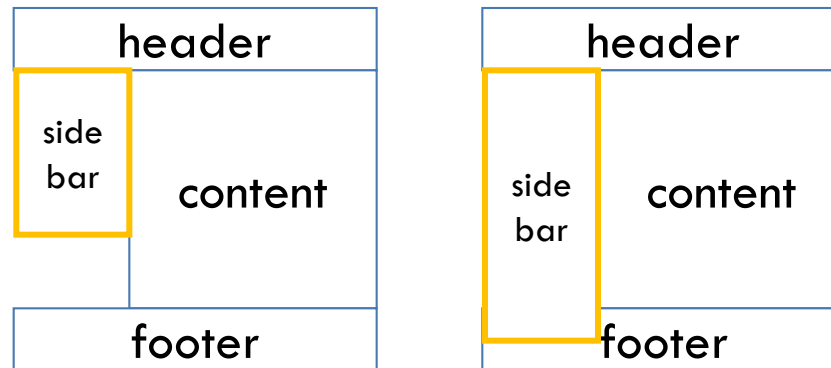
```
<body>  
  <header> ... </header>  
  <div id="sidebar"> ... </div>  
  <div id="content"> ... </div>  
  <footer> ... </footer>  
</body>
```



# Different lengths of columns

45

- Unless you fix the same height for both columns, they may have different height
  - ▣ You may paint the container background color appropriately to hide that fact that the sidebar that is too short.
  - ▣ If the sidebar is taller than content, it will overlap the footer.

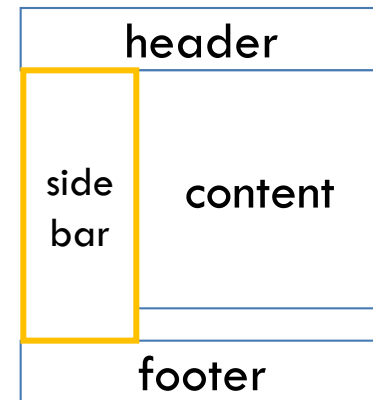


# Floating the sidebar

46

- A solution is to clear the footer from floats
  - ▣ Footer is always below content pane because of normal flow

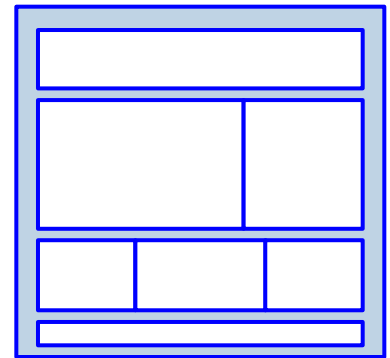
```
#sidebar { float: left; width: 150px; }  
#content { margin-left: 150px; }  
  
footer { clear: left; }
```



# Grid layout

47

- The web page is divided into several rows of the same width but different height
- Some row may be further divided into columns
  - We can use floating to position the boxes
  - CSS frameworks provide a convenient way to implement such layout.
    - Blueprint, <http://www.blueprintcss.org/>
    - 960 grid system, <http://960.gs/>
    - Bootstrap, <http://getbootstrap.com/>



# Laboratory

48

- Lab 4-5. Column layout
- Lab 4-6. Grid layout



# Part D. Flexible box model

49

- CSS3 introduces the **flexible box model**
  - ▣ Distribute boxes horizontally / vertically inside a container box
  - ▣ Boxes with flexible widths share remaining spaces in the container box
  - ▣ The boxes may be arranged in an order different from source order
  - ▣ Boxes in a row may stretch to the same height (similar to table cells in a table row)

# Flexbox is still a working draft

50

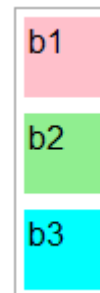
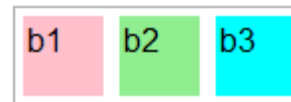
- The flexible box layout module is still a working draft
  - ▣ Not a standard yet
  - ▣ Firefox and WebKit based browsers (Safari, Chrome) have experimental implementation
- Reference:
  - ▣ [https://developer.mozilla.org/en-US/docs/Web/Guide/CSS/Flexible\\_boxes](https://developer.mozilla.org/en-US/docs/Web/Guide/CSS/Flexible_boxes)

# Using flexible box model

51

- Use `display: flex` to select flexible box model for a block element.
- Children in the block element will be arranged either horizontally or vertically
  - `flex-direction: row` (default)
  - `flex-direction: column`

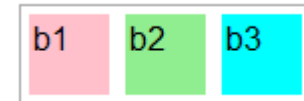
```
<div id="C">  
  <div id="b1"> ... </div>  
  <div id="b2"> ... </div>  
  <div id="b3"> ... </div>  
</div>
```



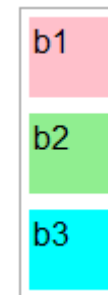
# Example

52

```
#C {  
  display: flex;  
  flex-direction: row;  
}  
#C div { width: 40px; height: 40px; }
```



```
#C {  
  display: flex;  
  flex-direction: column;  
}  
#C div { width: 40px; height: 40px; }
```



# Layout order

53

- You can reverse the order that children are laid out with **flex-direction: row-reverse or column-reverse**
- You can change the order that children are laid out by the property **order**
  - ▣ The box model will distribute boxes with order: 1 (default) first, then those with order: 2, and so on.

# Flexible box size

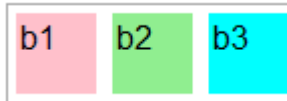
54

```
div#a1 { flex: 0 1 200px; }
```

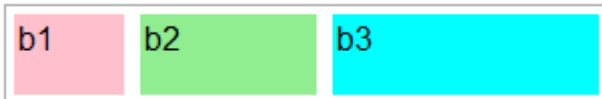
- The flex property specifies: flex-grow, flex-shrink and flex-basic. I.e. flexibility to grow, flexibility to shrink, and the basic size of a box
  - ▣ Default: `flex: 0 1 auto;` where auto refers to the width / height property.
- Allocates space in the container parent to children that have basic size set to a fixed values,
  - ▣ the remaining space is then allocated to each flexible child box in proportion to its flex-grow value.
  - ▣ If the container is not big enough, shrink the children in proportion to their flex-shrink value.

# Examples

55



```
#C { display: flex; width: 600px; }  
#b1, #b2, #b3 { flex: 1 1 auto; }
```



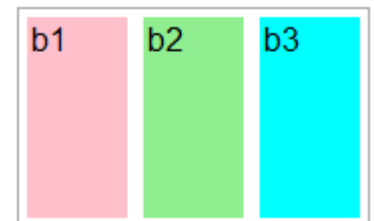
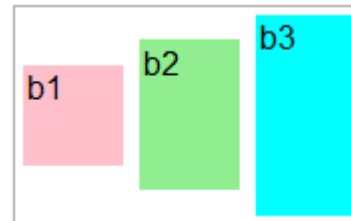
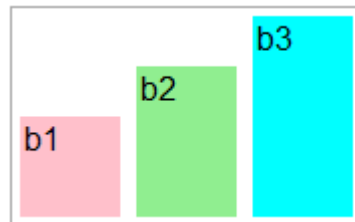
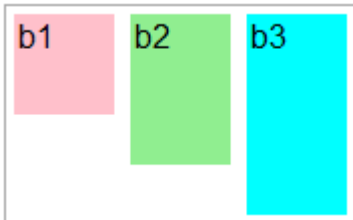
```
#C { display: flex; ; width: 300px; }  
#b1 { flex: 1 1 auto; }  
#b2 { flex: 2 1 auto; }  
#b3 { flex: 1 1 auto; width: 100px; }
```

# Distributing remaining vertical spaces

56

- (assuming box-direction: row)
- Height of boxes may be different and smaller than parent's height. How the vertical space is distributed on the vertical axis is determined by **align-items**

<b>align-items: flex-start</b>	align at the top of the parent
<b>align-items: flex-end</b>	align at the bottom of the parent
<b>align-items: center</b>	each box is placed at the center
<b>align-items: stretch</b>	height of each box is adjusted to fit the height of the parent





# Distributing remaining horizontal spaces

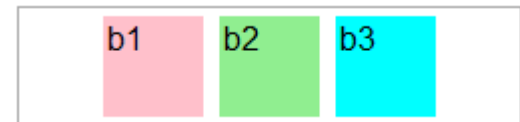
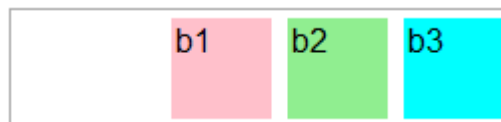
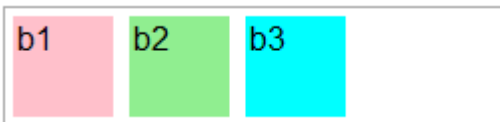
57

- (assuming box-orient: horizontal)
- The total width of child boxes may be smaller than the parent's width. How the remaining horizontal space is distributed is determined by **justify-content**

<b>justify-content: flex-start</b>	child boxes are pushed to the left
------------------------------------	------------------------------------

<b>justify-content: flex-end</b>	child boxes are pushed to the right
----------------------------------	-------------------------------------

<b>justify-content: center</b>	child boxes are placed at center
--------------------------------	----------------------------------



# Further readings

58

- A wiki about CSS: [http://css-discuss.incutio.com/wiki/Main\\_Page](http://css-discuss.incutio.com/wiki/Main_Page)
- CSS codes and library: <http://www.dynamicdrive.com/style/>
- Layout:
  - Layout gallery: <http://blog.html.it/layoutgala/>
  - Layout tutorial:  
[http://www.maxdesign.com.au/presentation/page\\_layouts/](http://www.maxdesign.com.au/presentation/page_layouts/)
- The new layout in DW CS5  
[http://www.adobe.com/devnet/dreamweaver/articles/introducing\\_new\\_css\\_layouts.html](http://www.adobe.com/devnet/dreamweaver/articles/introducing_new_css_layouts.html)