Introduction to Web Programming

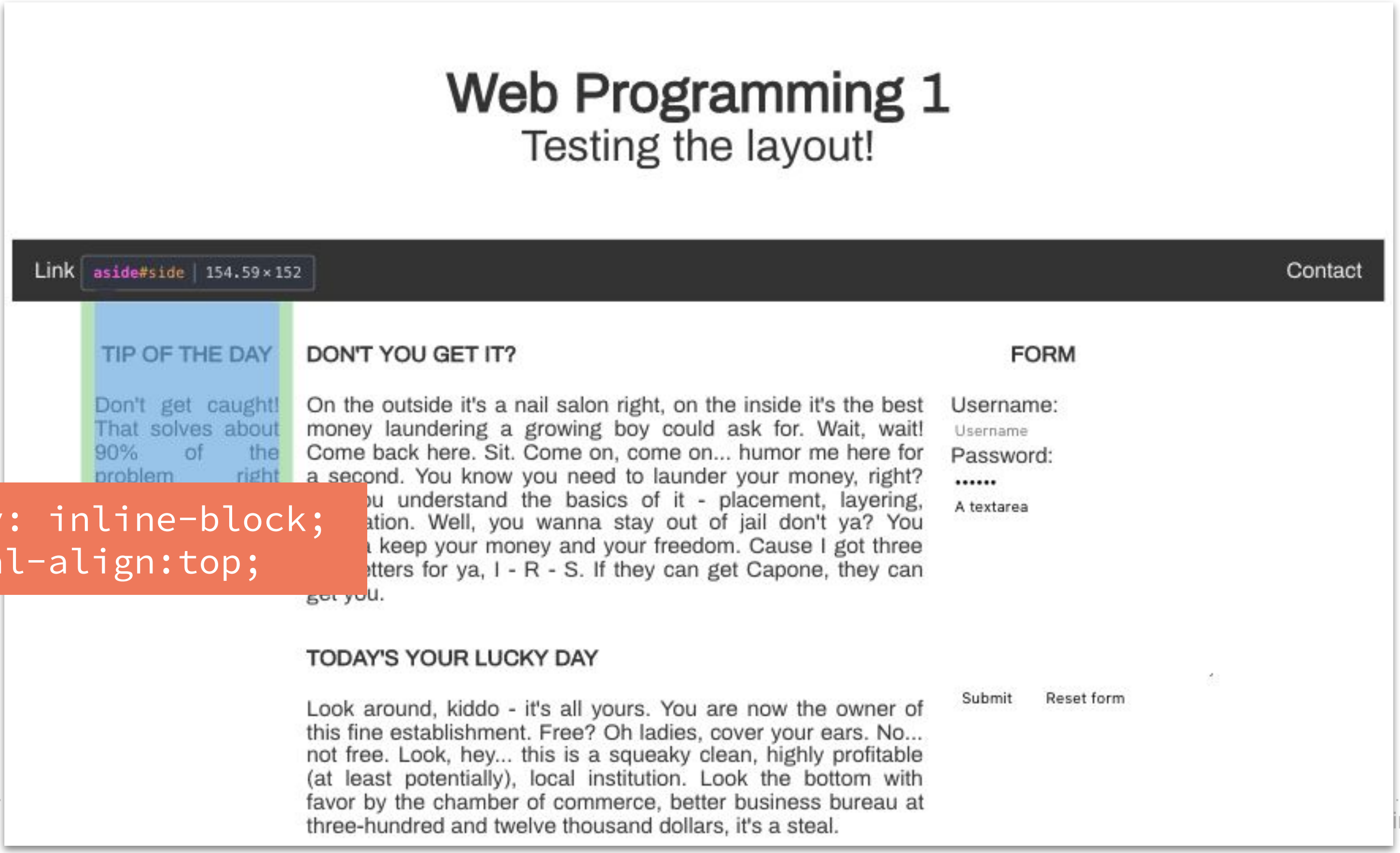
Chap. 3 / Responsive Web Design with CSS3

Anne Jeannin-Girardon, PhD | anne.jeannin@unistra.fr
Associate Professor, University of Strasbourg

Today’s topics

- Laying out elements with the property display
- Towards responsive designs
- Introduction to the Flexbox model
- Media queries

Application: webpage layout



Web Programming 1

Testing the layout!

Link aside#side 154.59 × 152 Contact

TIP OF THE DAY

Don't get caught! That solves about 90% of the problem right

DON'T YOU GET IT?

On the outside it's a nail salon right, on the inside it's the best money laundering a growing boy could ask for. Wait, wait! Come back here. Sit. Come on, come on... humor me here for a second. You know you need to launder your money, right? ...u understand the basics of it - placement, layering, ...ation. Well, you wanna stay out of jail don't ya? You ... keep your money and your freedom. Cause I got three ...tters for ya, I - R - S. If they can get Capone, they can get you.

TODAY'S YOUR LUCKY DAY

Look around, kiddo - it's all yours. You are now the owner of this fine establishment. Free? Oh ladies, cover your ears. No... not free. Look, hey... this is a squeaky clean, highly profitable (at least potentially), local institution. Look the bottom with favor by the chamber of commerce, better business bureau at three-hundred and twelve thousand dollars, it's a steal.

FORM

Username:

Username

Password:

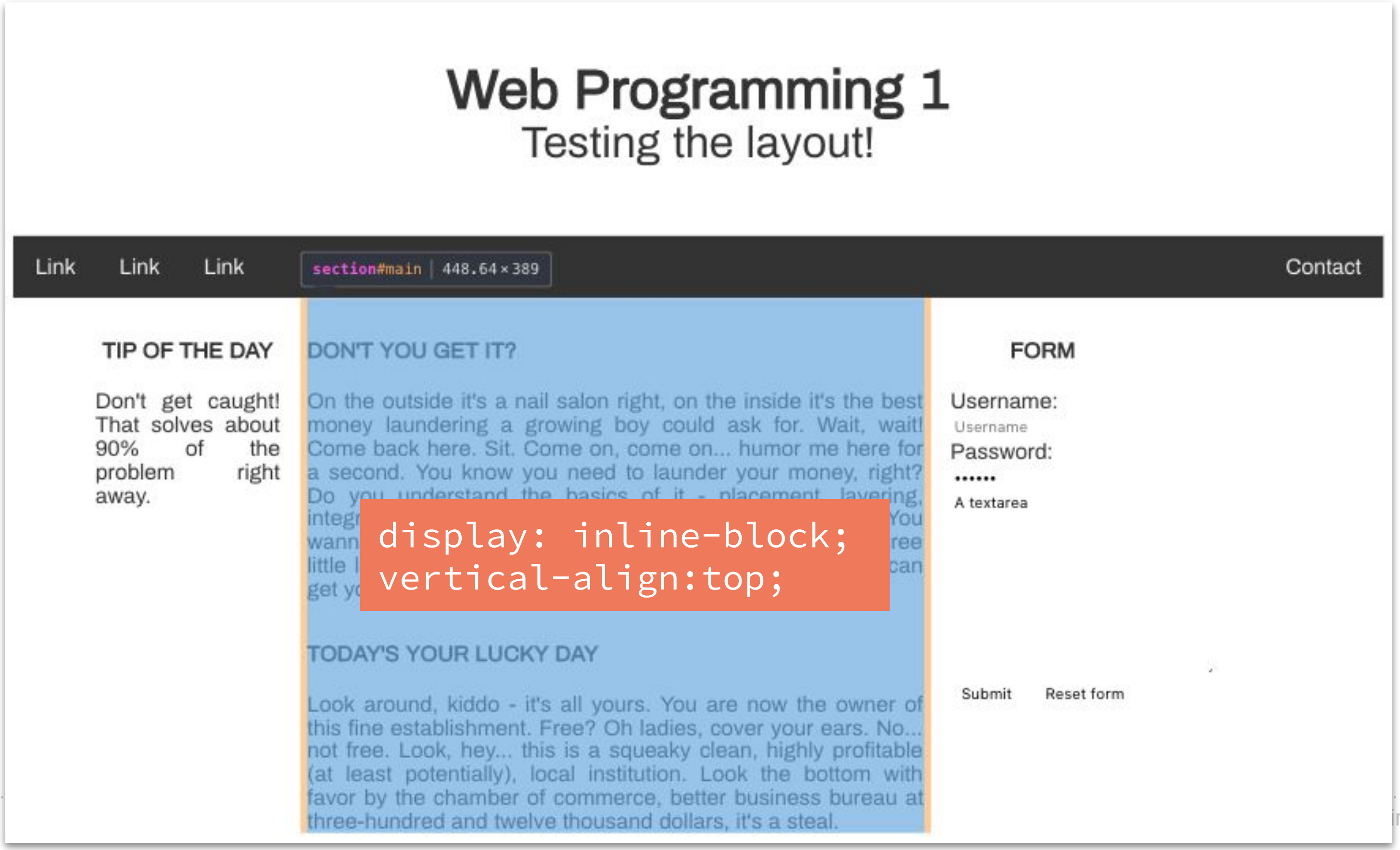
••••••

A textarea

Submit Reset form

```
display: inline-block;
vertical-align:top;
```

Application: webpage layout



Web Programming 1

Testing the layout!

TIP OF THE DAY

Don't get caught! That solves about 90% of the problem right away.

DON'T YOU GET IT?

On the outside it's a nail salon right, on the inside it's the best money laundering a growing boy could ask for. Wait, wait! Come back here. Sit. Come on, come on... humor me here for a second. You know you need to launder your money, right? Do you understand the basics of it - placement, layering, integr… You wann… ree little l… can get yo…

TODAY'S YOUR LUCKY DAY

Look around, kiddo - it's all yours. You are now the owner of this fine establishment. Free? Oh ladies, cover your ears. No... not free. Look, hey... this is a squeaky clean, highly profitable (at least potentially), local institution. Look the bottom with favor by the chamber of commerce, better business bureau at three-hundred and twelve thousand dollars, it's a steal.

FORM

Username:

Username

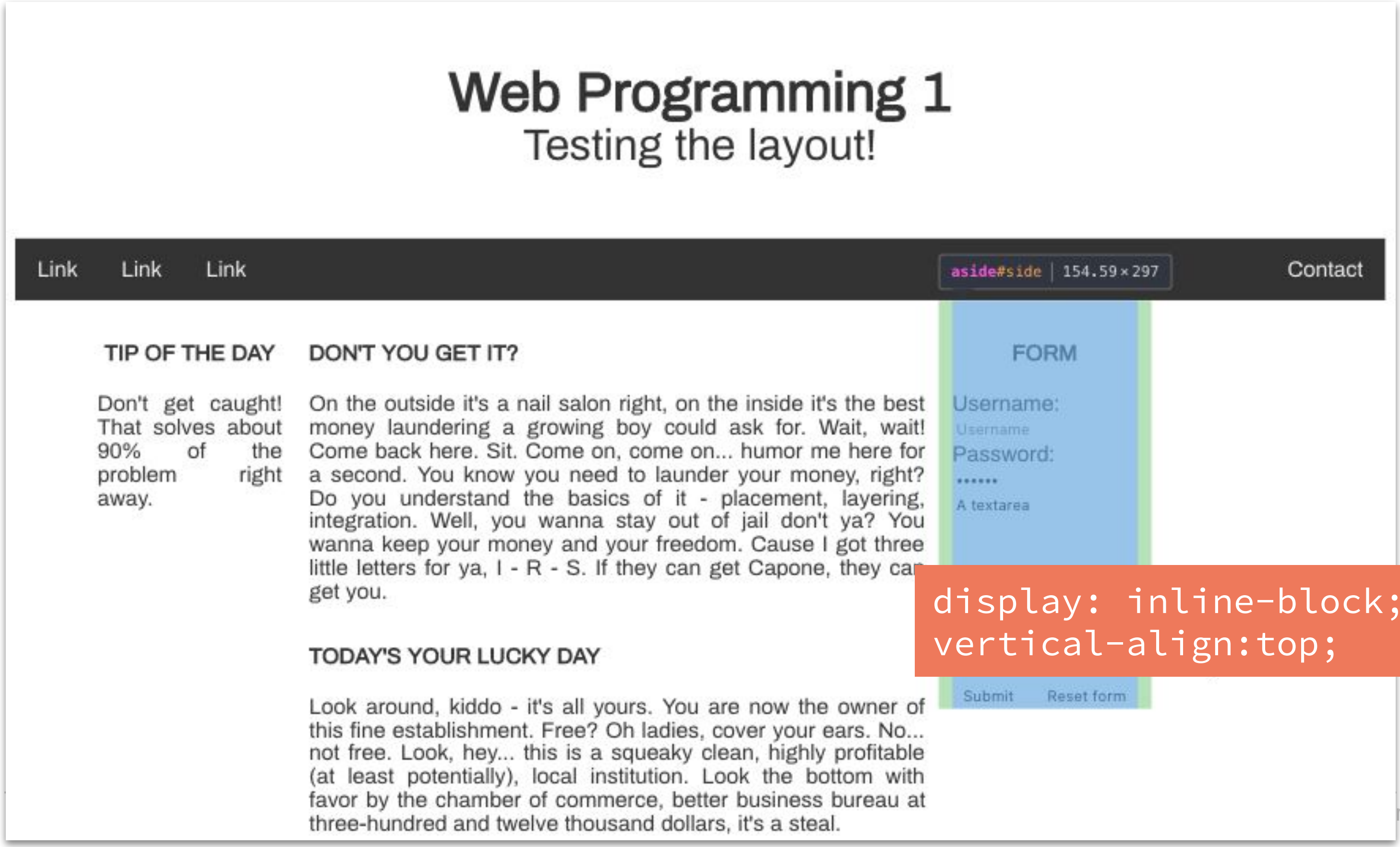
Password:

••••••

A textarea

```
display: inline-block;
vertical-align:top;
```

Application: webpage layout



Web Programming 1

Testing the layout!

TIP OF THE DAY

Don't get caught! That solves about 90% of the problem right away.

DON'T YOU GET IT?

On the outside it's a nail salon right, on the inside it's the best money laundering a growing boy could ask for. Wait, wait! Come back here. Sit. Come on, come on... humor me here for a second. You know you need to launder your money, right? Do you understand the basics of it - placement, layering, integration. Well, you wanna stay out of jail don't ya? You wanna keep your money and your freedom. Cause I got three little letters for ya, I - R - S. If they can get Capone, they ca get you.

TODAY'S YOUR LUCKY DAY

Look around, kiddo - it's all yours. You are now the owner of this fine establishment. Free? Oh ladies, cover your ears. No... not free. Look, hey... this is a squeaky clean, highly profitable (at least potentially), local institution. Look the bottom with favor by the chamber of commerce, better business bureau at three-hundred and twelve thousand dollars, it's a steal.

FORM

Username:

Username

Password:

••••••

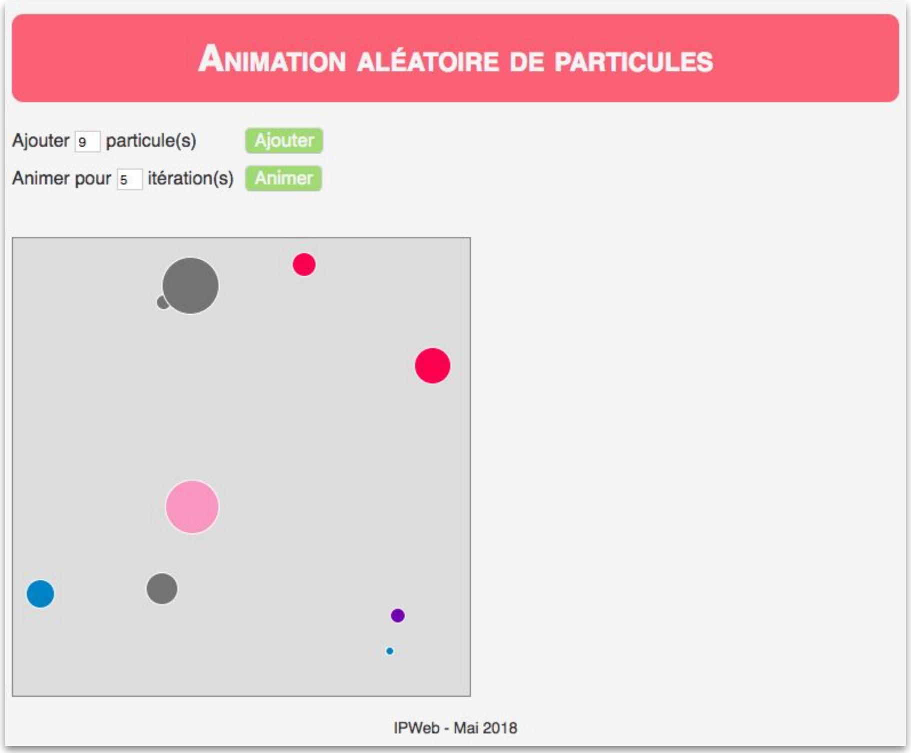
A textarea

Submit Reset form

```
display: inline-block;
vertical-align:top;
```

Application: animate particles

- Container position for the particules set as relative
- Particles position set as absolute (constrained to the container zone)
- Particles displayed as inline-block



ANIMATION ALÉATOIRE DE PARTICULES

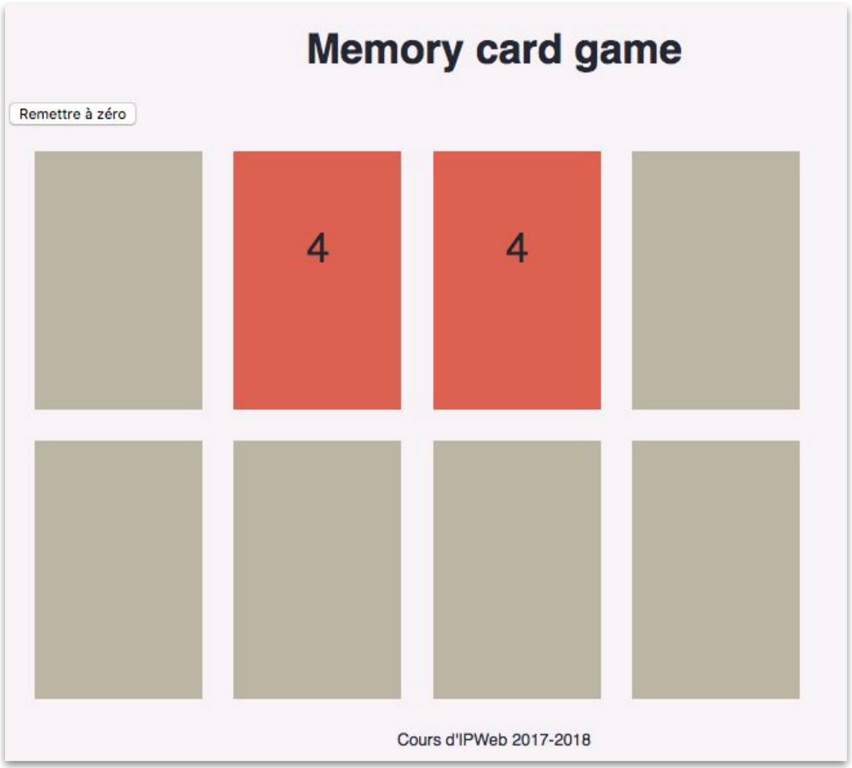
Ajouter 9 particule(s) Ajouter

Animer pour 5 itération(s) Animer

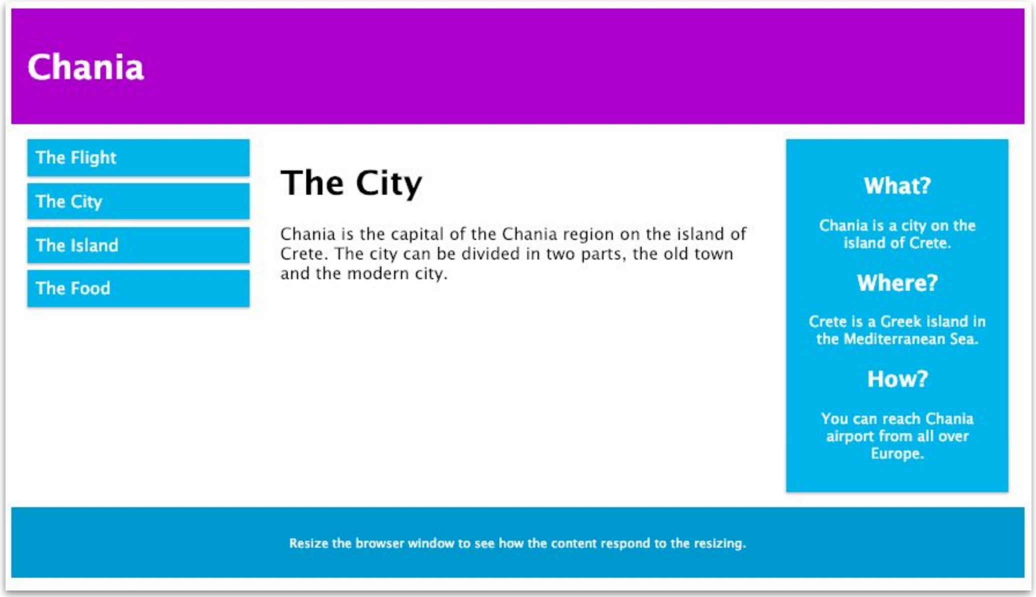
IPWeb - Mai 2018

Application: memory card game

- Card displayed as inline-block
- Front and back are displayed as absolute (parent: card)
- Front is displayed at none by default (must be revealed by the player)



Responsive Web Design (RWD)



Chania

The Flight
The City
The Island
The Food

The City

Chania is the capital of the Chania region on the island of Crete. The city can be divided in two parts, the old town and the modern city.

What?

Chania is a city on the island of Crete.

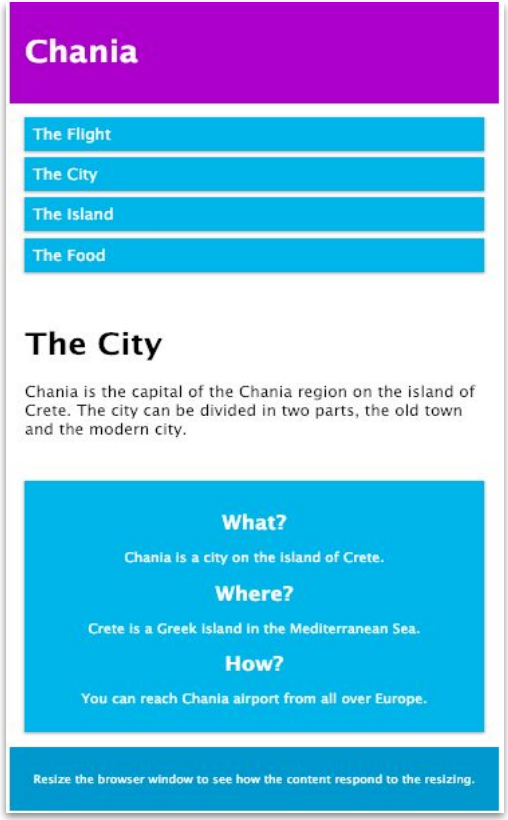
Where?

Crete is a Greek island in the Mediterranean Sea.

How?

You can reach Chania airport from all over Europe.

Resize the browser window to see how the content respond to the resizing.



Chania

The Flight
The City
The Island
The Food

The City

Chania is the capital of the Chania region on the island of Crete. The city can be divided in two parts, the old town and the modern city.

What?

Chania is a city on the island of Crete.

Where?

Crete is a Greek island in the Mediterranean Sea.

How?

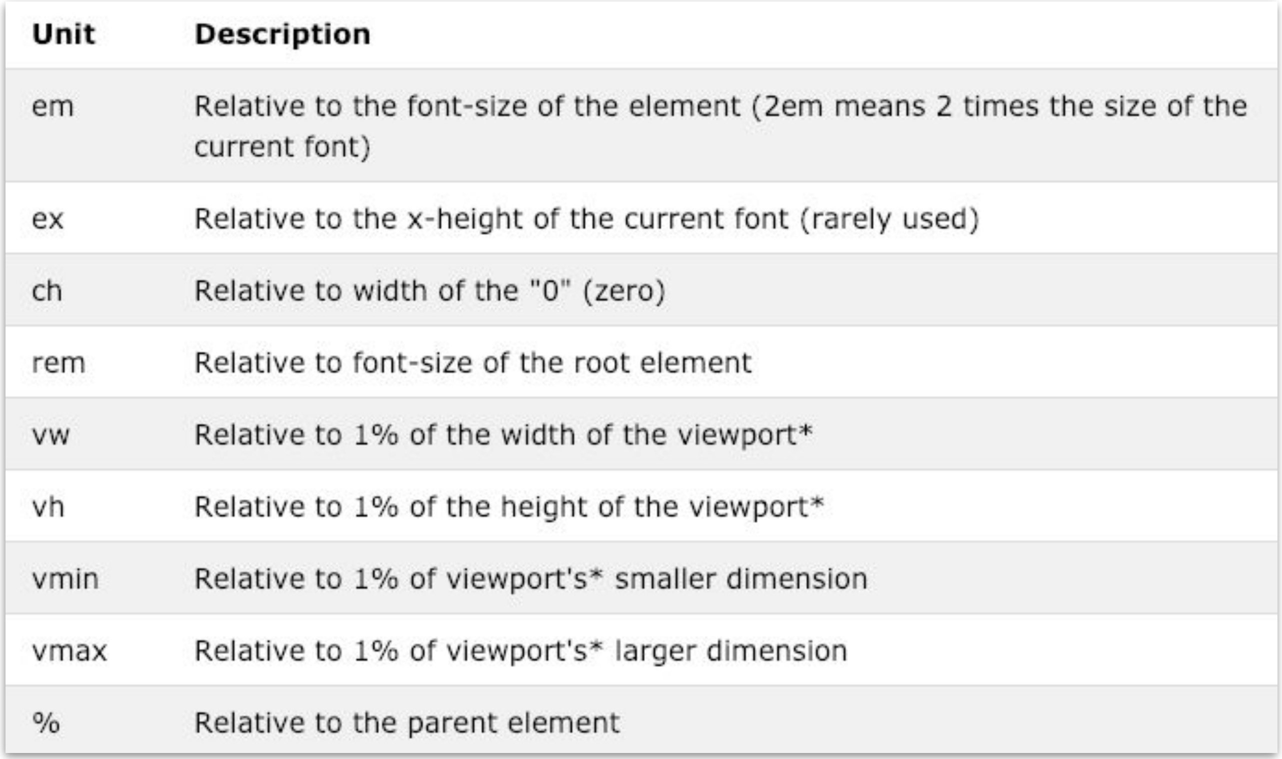
You can reach Chania airport from all over Europe.

Resize the browser window to see how the content respond to the resizing.

Responsive Web Design

- How is the content displayed across various devices (laptops, mobile phones, tablets, …)
- Give the user the best possible experience of your web page
- Only one style sheet ! just use a few technologies to build a flexible layout
  - Fluid layout (Flexbox, CSS Grid, Bootstrap grid)
  - Relative sized media, fonts, etc. (i.e. no fixed pixel sizes)
  - Media queries

CSS relative sizes



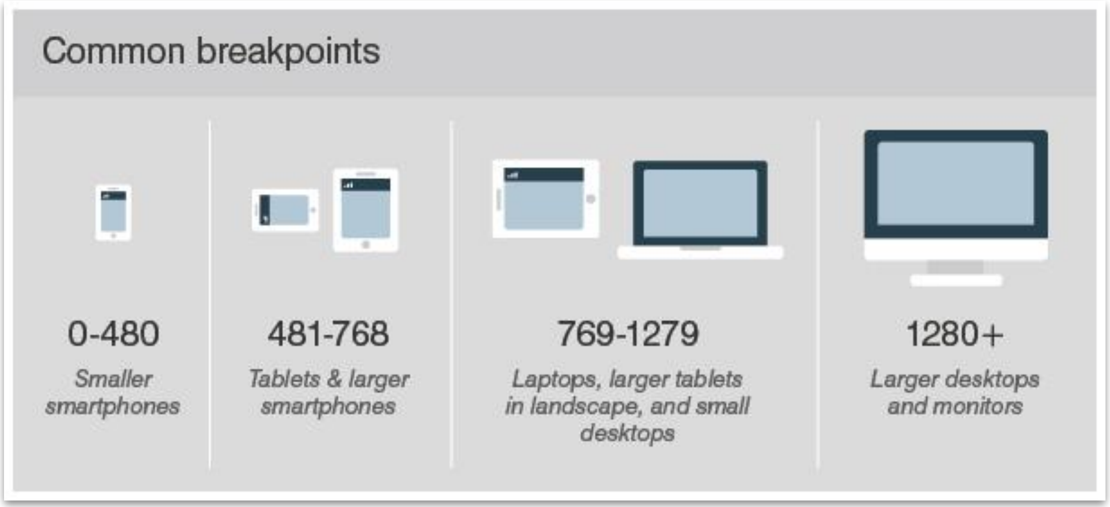
Unit	Description
em	Relative to the font-size of the element (2em means 2 times the size of the current font)
ex	Relative to the x-height of the current font (rarely used)
ch	Relative to width of the "0" (zero)
rem	Relative to font-size of the root element
vw	Relative to 1% of the width of the viewport*
vh	Relative to 1% of the height of the viewport*
vmin	Relative to 1% of viewport's* smaller dimension
vmax	Relative to 1% of viewport's* larger dimension
%	Relative to the parent element

Elements of RWD

- Controlling the viewport

```
<meta name="viewport" content="width=device-width, initial-scale=1">
```

  - Ask the browser to render the width of the page to the same width of the browser window
- Using a grid system (e.g. flexbox)
- Using relative sizes (%, em, ...)
- Setting appropriate breakpoints for media queries


Common breakpoints
0-480
Smaller smartphones
481-768
Tablets & larger smartphones
769-1279
Laptops, larger tablets in landscape, and small desktops
1280+
Larger desktops and monitors

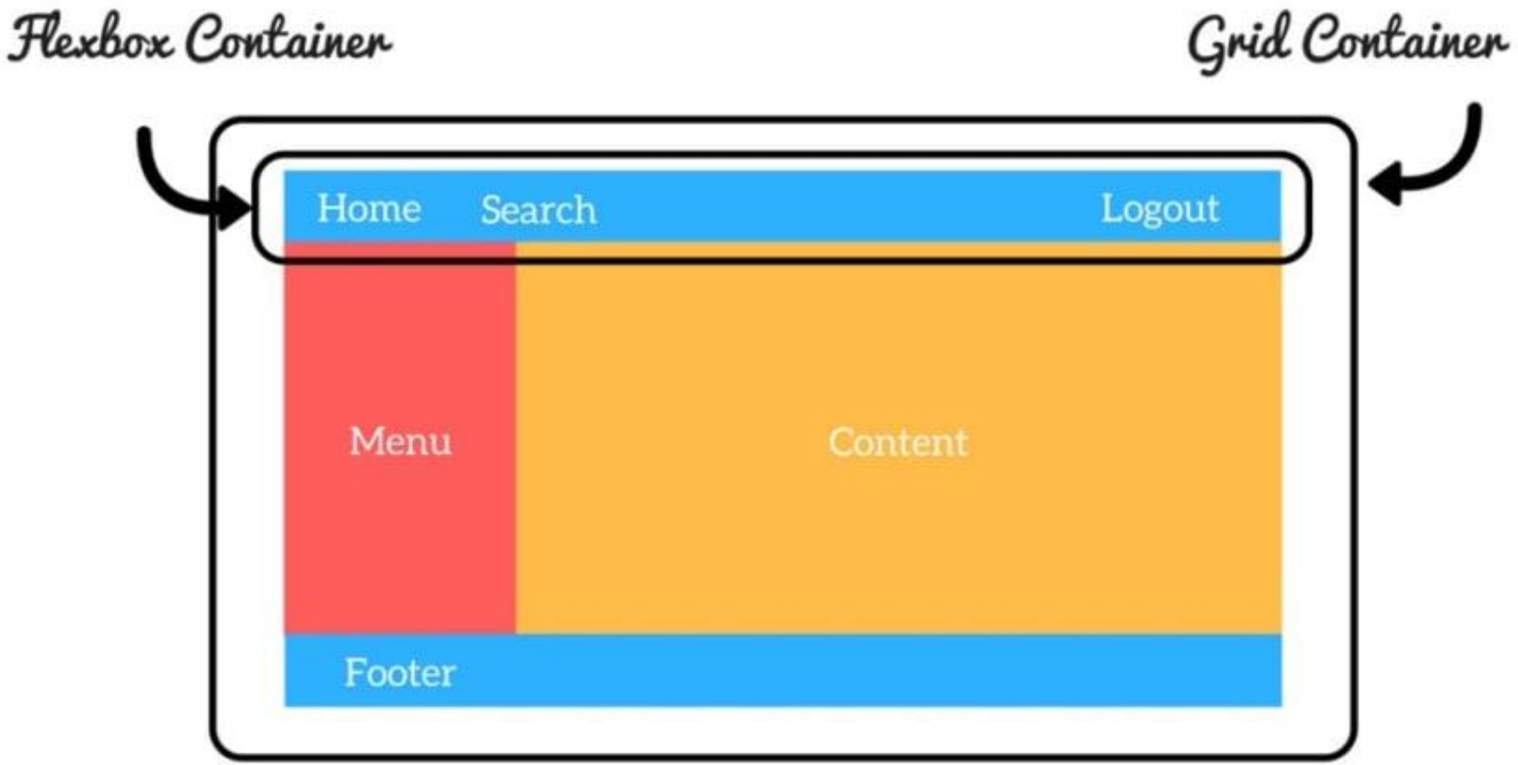
Src: A. Libby, RWD w/ HTML5 and CSS3

Flexbox or CSS grid ?

- Flexbox :
  - 1D containers (lay out elements either in a row or in a column)
  - content oriented (create the layout after you have set up your content)
- CSS Grid :
  - 2D containers (lay out elements in both in rows and columns)
  - layout oriented (think the layout first then add content)
  - fits better to larger scale layouts

We will focus on Flexbox

(a good starting point, maybe a bit easier / once Flexbox is mastered, it’s not difficult to learn CSS grid)



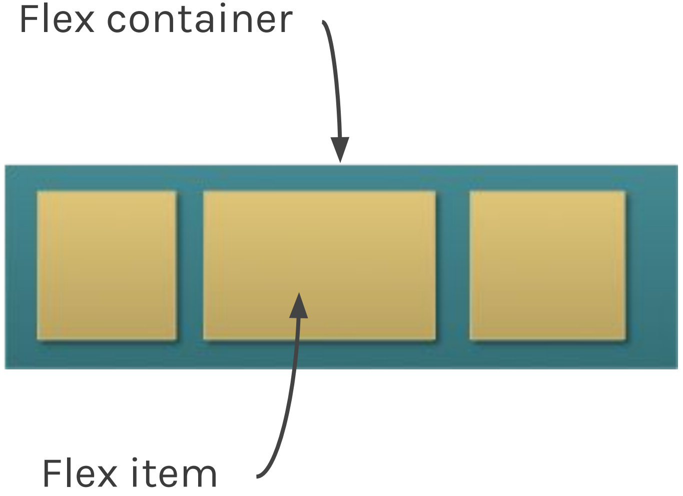
https://medium.com/youstart-labs/beginners-guide-to-choose-between-css-grid-and-flexbox-783005dd2412

Flexbox : FLEXible BOX

- Main idea : fit the available space the best possible way (elements can change their dimensions)
- So it takes into account a change of orientation or a different viewport
- The notion of normal flow no longer applies
- But the notion of parent/children still does !

Flexbox parent/children

- Parents : flex containers
- Children : flex items
- Different properties apply either to containers, items (or both)

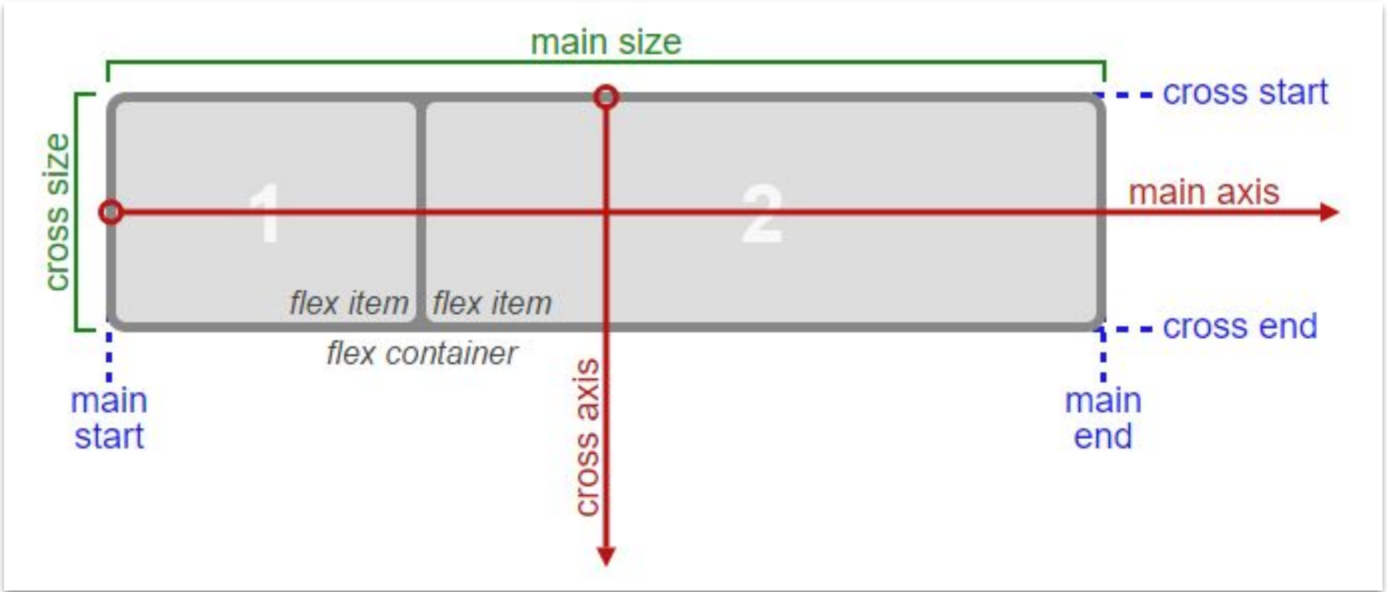


Flexbox and flow

In order to display the elements in the page, we will look at how elements can be :

- Distributed (horizontal / vertical ; wrapping)
- Aligned (centered, justified, stretched horizontally or vertically)
- Organised (independently from the order they have in the HTML code)

Main and secondary axis



Defining a flex container

- All the children of this element will be in a flex context
- If you want the container to be displayed as an inline element, use the property `inline-flex`

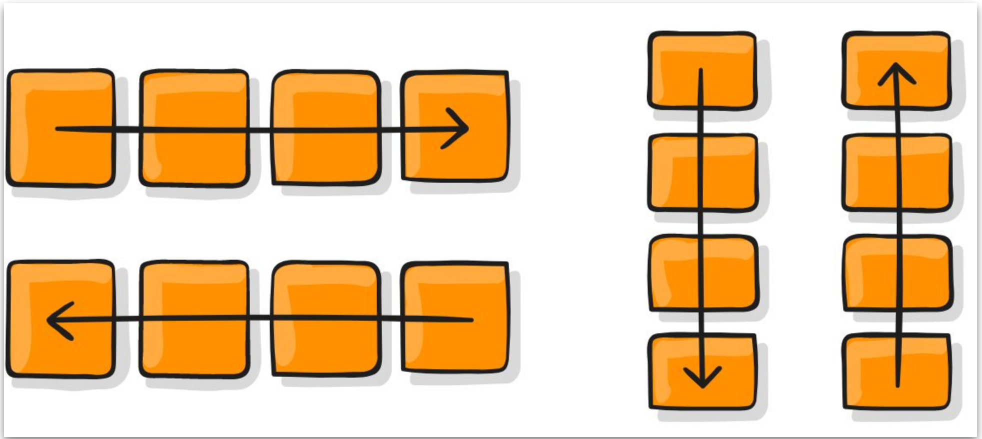
Flexbox

```css
.container {
  display: flex;
  /* or inline-flex */
}
```

Container properties (1/6)

`flex-direction`

Sets the main axis



Illustrations from csstricks.com

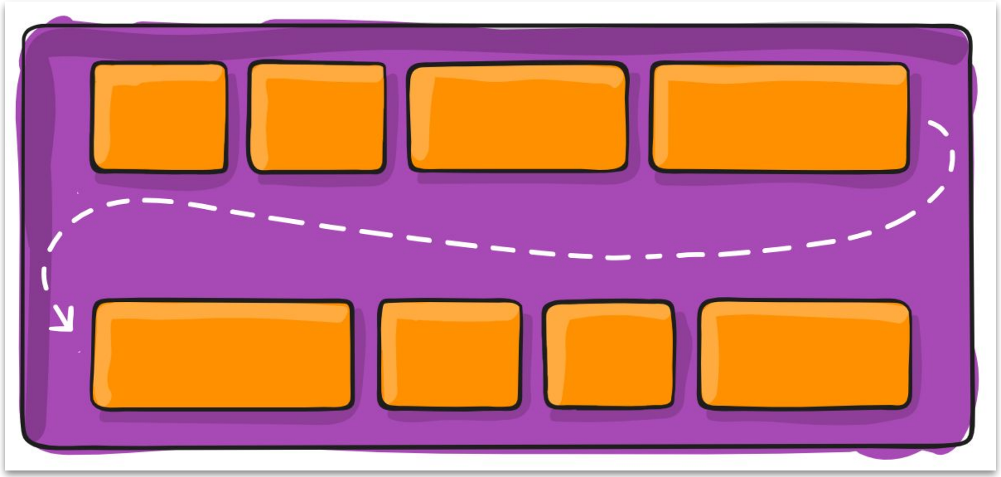
Flexbox

```
.container {
  display: flex;
  /* or inline-flex */
  flex-direction: row;
  /* or row-reverse, column,
     column-reverse */
}
```

Container properties (2/6)

`flex-wrap`

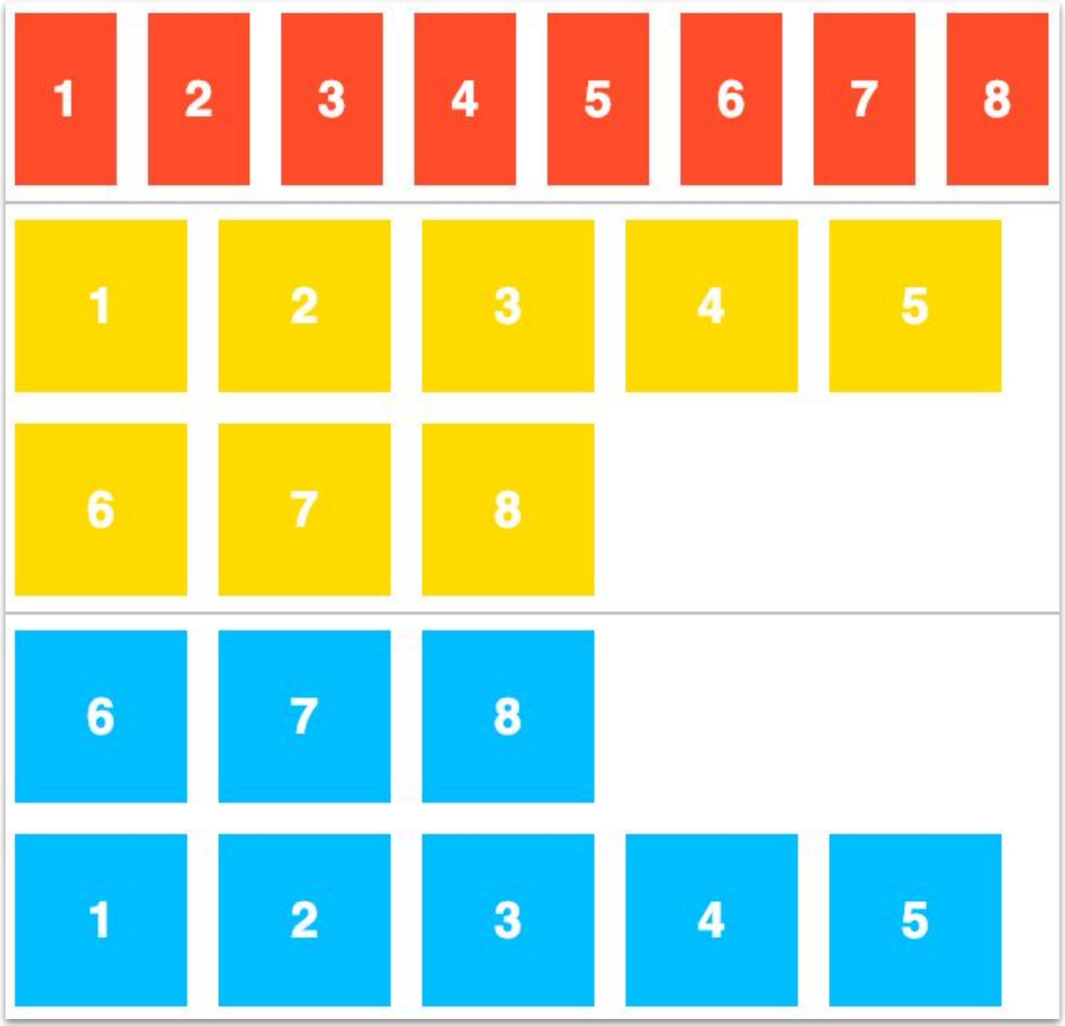
Allows items to wrap if they don't fit onto one single line



Illustrations from csstricks.com

Flexbox

```
.container {
  /* ... */
  flex-wrap: nowrap;
  /* or wrap, wrap-reverse */
}
```


1
2
3
4
5
6
7
8
1
2
3
4
5
6
7
8
6
7
8
1
2
3
4
5

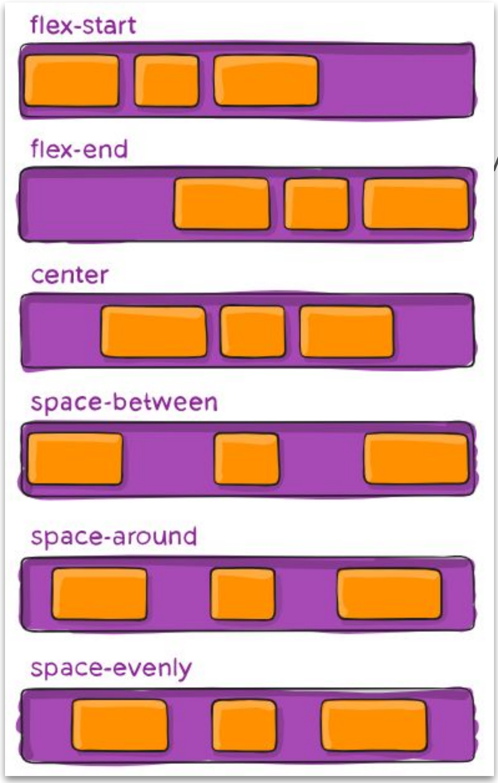
Container properties (3/6)

Use `flex-flow` to specify both direction and wrapping at the same time

Flexbox

```
.container {
  /* ... */
  flex-flow: <direction> <wrap>;
}
```

Container properties (4/6)



Illustrations from csstricks.com

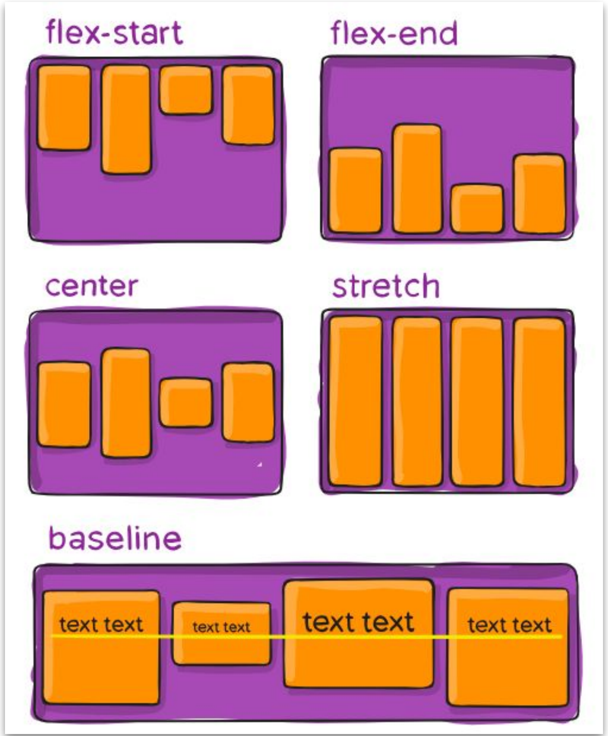
`justify-content`

Alignment along the <u>main axis</u> ; helps distributing the space

Flexbox

```
.container {
  /* ... */
  justify-content: flex-start;
  /* or flex-end, center,
    space-between, space-around,
    space evenly */
}
```

Container properties (5/6)



Illustrations from csstricks.com

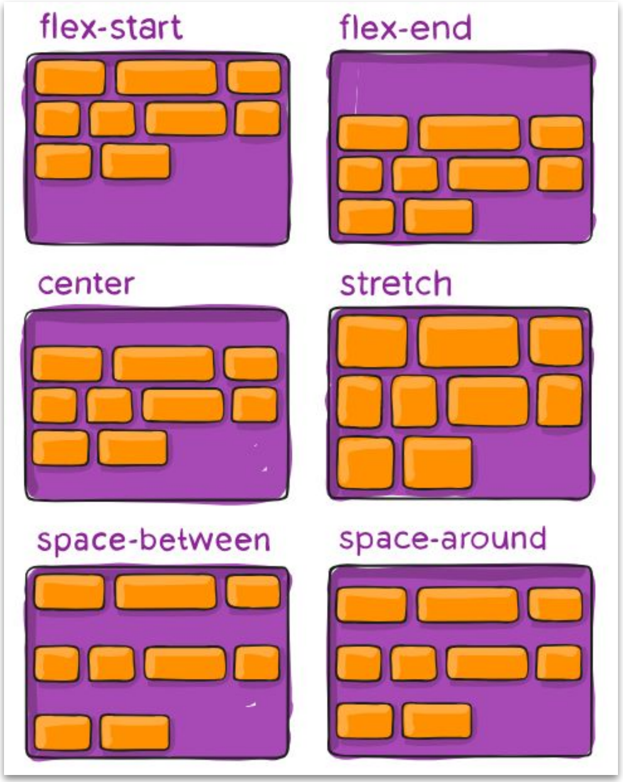
`align-items`

Alignment along the secondary axis

Flexbox

```
.container {
  /* ... */
  align-items: stretch;
  /* or flex-start, flex-end,
     center, baseline */
}
```

Container properties (6/6)



Illustrations from csstricks.com

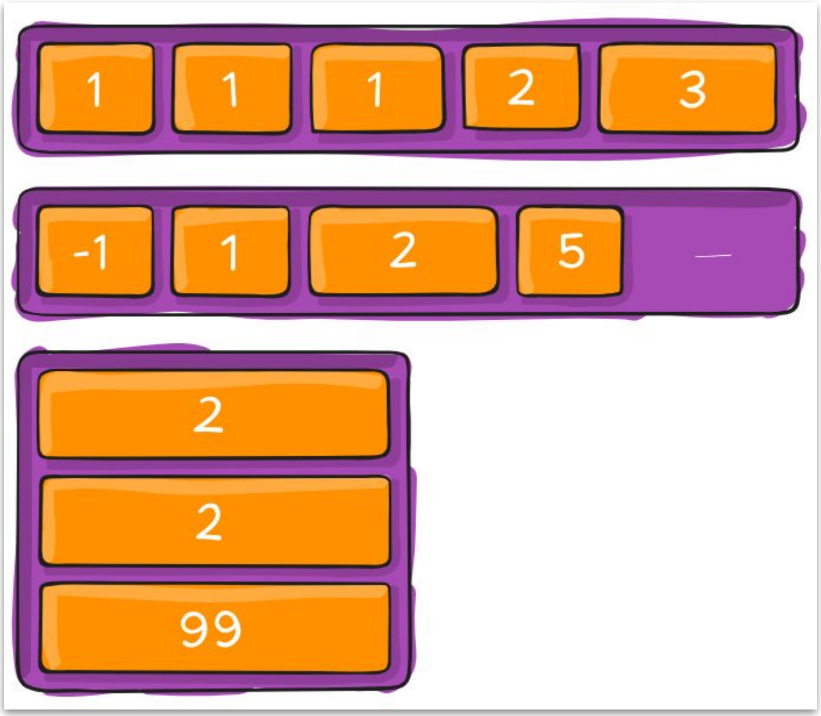
align-content

Distribute space on the lines of the container

Flexbox

```
.container {
  /* ... */
  align-content: stretch;
  /* or flex-start, flex-end,
     center, space-between,
     space-around */
}
```

Item properties (1/4)



Illustrations from csstricks.com

order

Controls the order in which items appear in the container (may be different from the order in the HTML structure !)

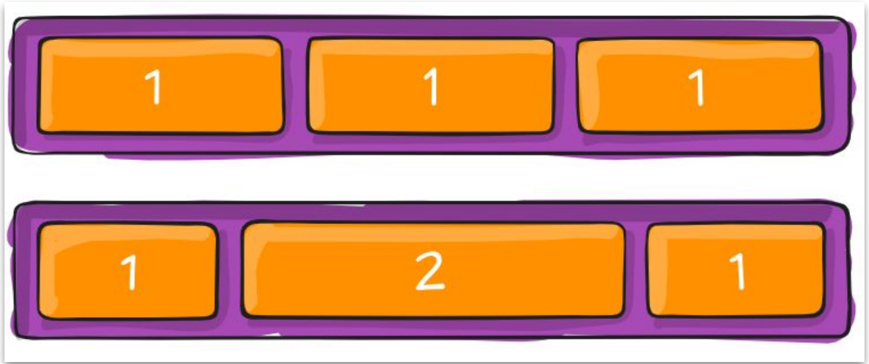
Flexbox

```
.item {
  /* ... */
  order: 0;
  /* 0: default */
}
```

Item properties (2/4)

`flex-grow` `flex-skrink`

Ability of an item to grow / shrink (proportion)



Flexbox

```
.itemA {
  flex-grow: 1;
}

.itemB {
  flex-grow: 2;
  /* tries to take twice as much
     space as other items */
}
```

Illustrations from csstricks.com

Item properties (3/4)

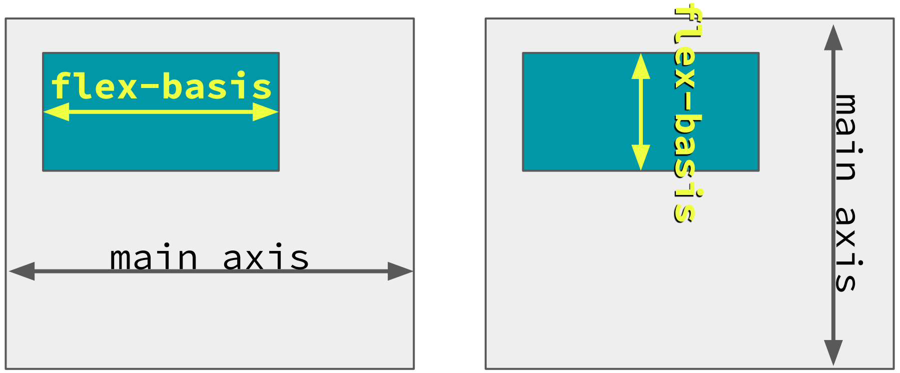
`flex-basis`

Default width (in the main axis) of an item before the remaining space is distributed in the container

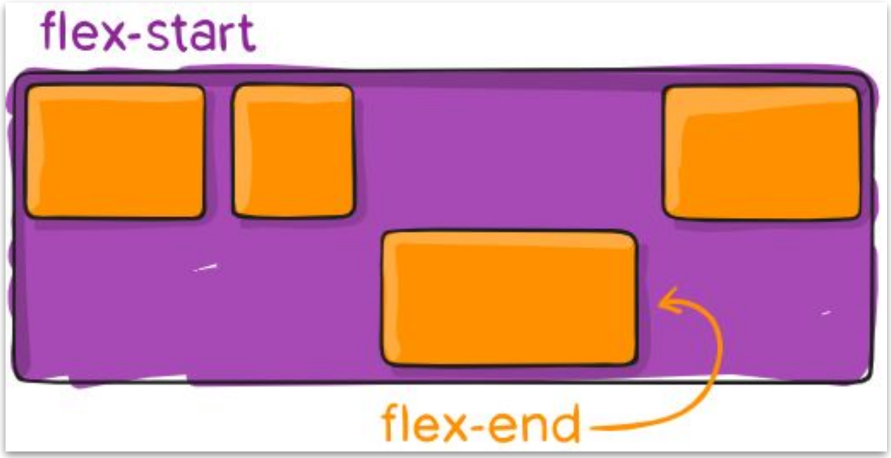
Roughly equivalent to **`width`** --for an inline-block element-- but has the priority over it and it's relative to the main axis (whereas width is just ↔)

`flex`

Shorthand for flex-grow, flex-shrink (opt.) and flex-basis (opt.)



Item properties (4/4)



Illustrations from csstricks.com

align-self

Defines (overrides) the default container alignment at the level of individual items

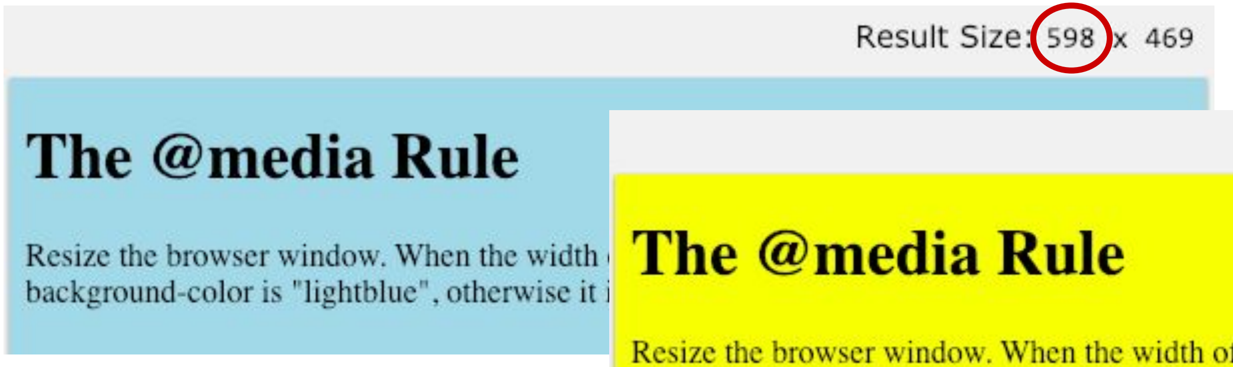
Flexbox

```
.container {
  align-items: flex-start;
}

.item {
  align-self: flex-end;
}
```

Media queries

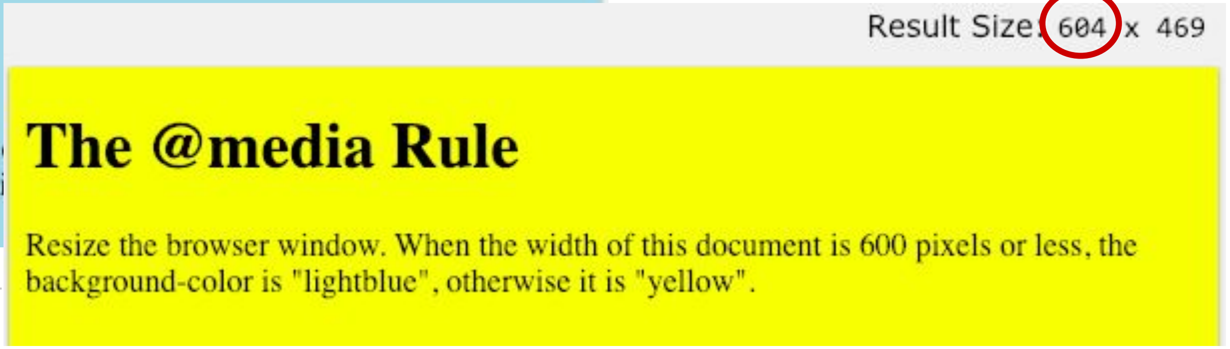
- Main idea : execute CSS code under certain conditions
- Use the keyword `@media` in your CSS, select a medium (all, screen, print, speech), apply a condition (e.g. min-width: 480px)

```
body {
  background-color: yellow;
}

@media screen and (max-width: 600px) {
  body {
    background-color: lightblue;
  }
}
```


Result Size: 598 x 469

The @media Rule

Resize the browser window. When the width
background-color is "lightblue", otherwise it i

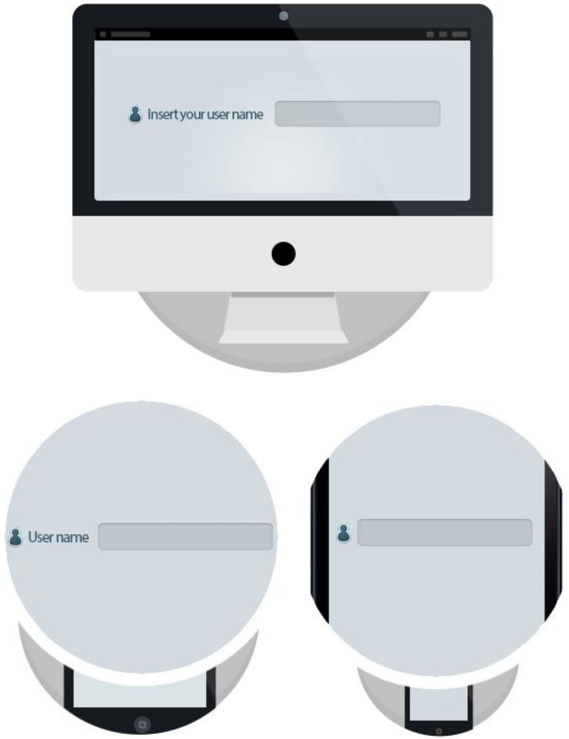

Result Size: 604 x 469

The @media Rule

Resize the browser window. When the width of this document is 600 pixels or less, the background-color is "lightblue", otherwise it is "yellow".

Media queries, another example

```
.username:after {
    content:"Insert your user name";
}
@media screen and (max-width: 1024px) {
    .username:before {
        content:"User name";
    }
}
@media screen and (max-width: 480px) {
    .username:before {
        content:"";
    }
}
```



Illustrations from toptal.com

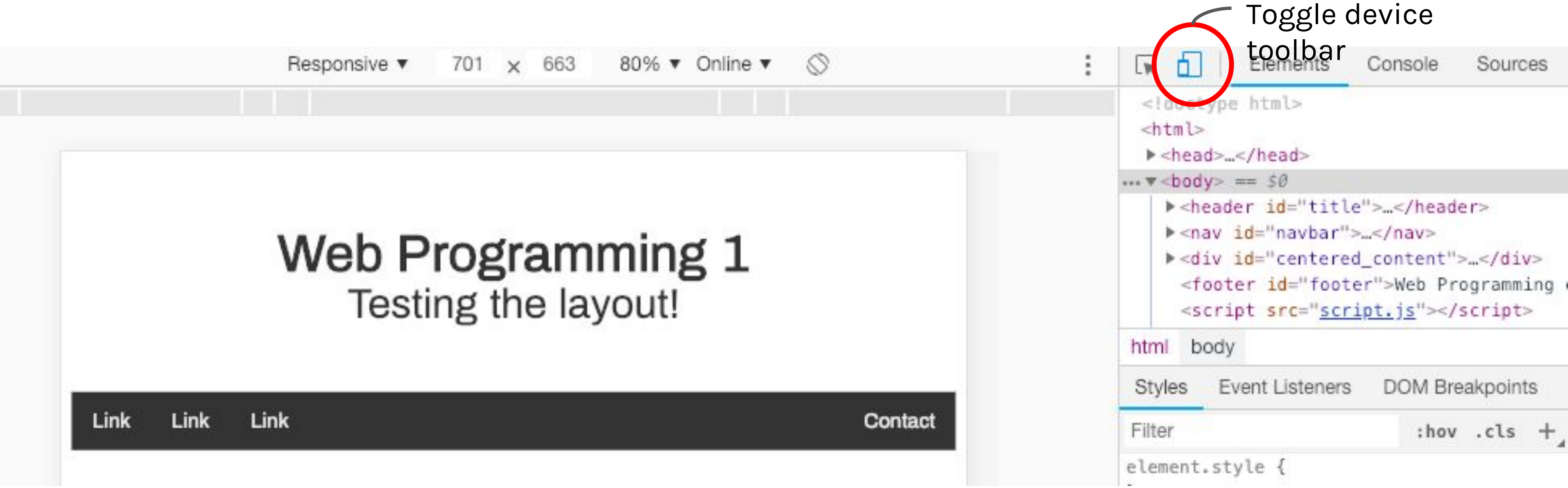
Media queries : breakpoints

```css
/* Extra small devices (phones, 600px and down) */
@media screen and (max-width: 600px) {...}

/* Small devices (portrait tablets and large phones, 600px and up) */
@media screen and (min-width: 600px) {...}

/* Medium devices (landscape tablets, 768px and up) */
@media screen and (min-width: 768px) {...}

/* Large devices (laptops/desktops, 992px and up) */
@media screen and (min-width: 992px) {...}

/* Extra large devices (large laptops and desktops, 1200px and up) */
@media screen and (min-width: 1200px) {...}
```

Go mobile first !

- A piece of advice you'll often see is : build your mobile layout first (it's easier to go from a highly constrained medium to a less constrained one)
- Then, add media queries to specify desktop layout and behavior
- min-width : mobile first (regular css are the properties for small screens)
- max-width : desktop first

Testing media using developer tools



Web Programming 1
Testing the layout!

```
<!doctype html>
<html>
 ▶<head>…</head>
…▼<body> == $0
  ▶<header id="title">…</header>
  ▶<nav id="navbar">…</nav>
  ▶<div id="centered_content">…</div>
   <footer id="footer">Web Programming
   <script src="script.js"></script>
```

```
element.style {
```

Chapter recap

- Use CSS display to build a page layout
- Introduction to the concepts behind RWD
- The Flexbox model
- Media Queries for RWD