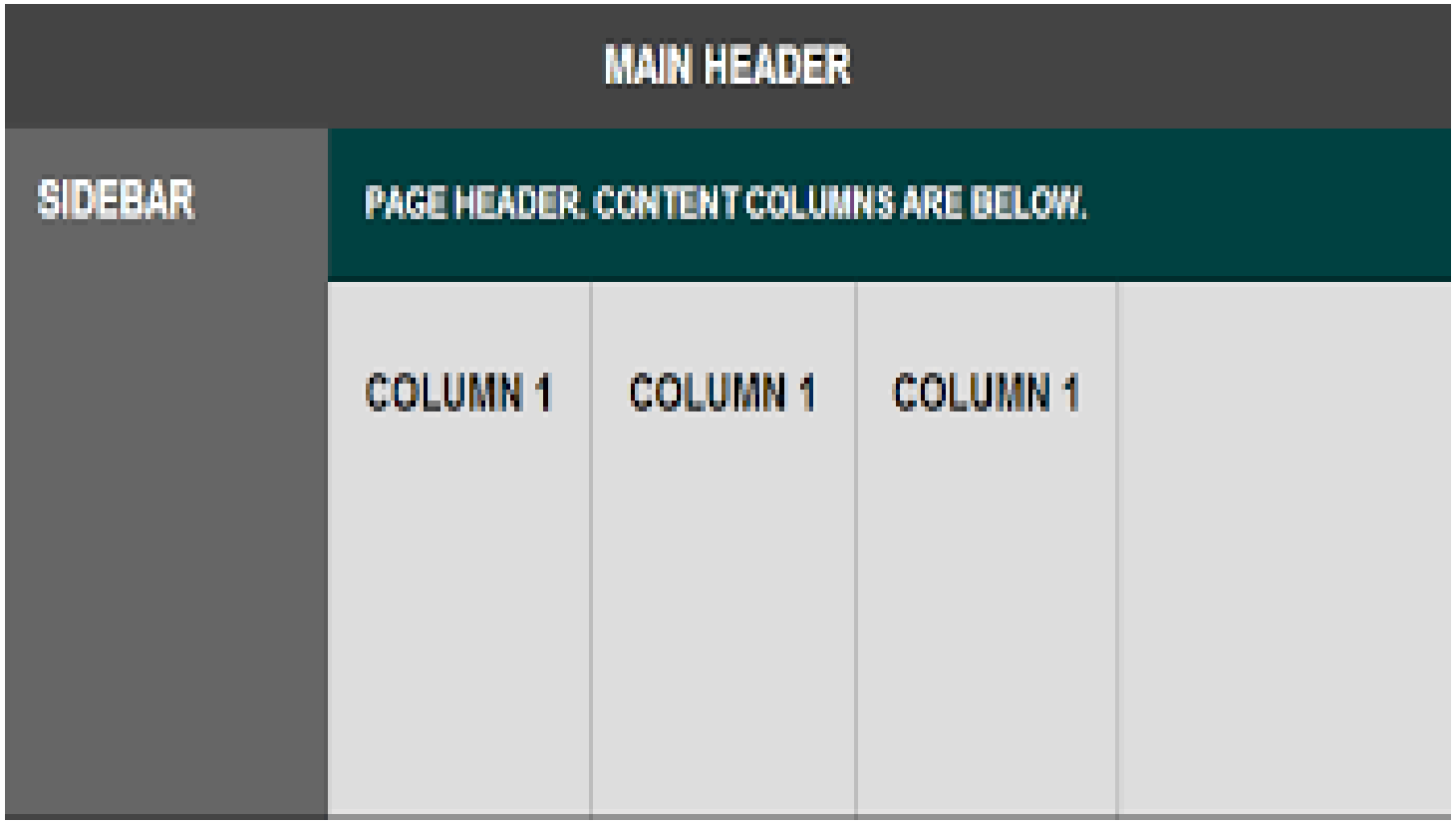
Web Development Funamentals



Lecturer: Marie Brennan

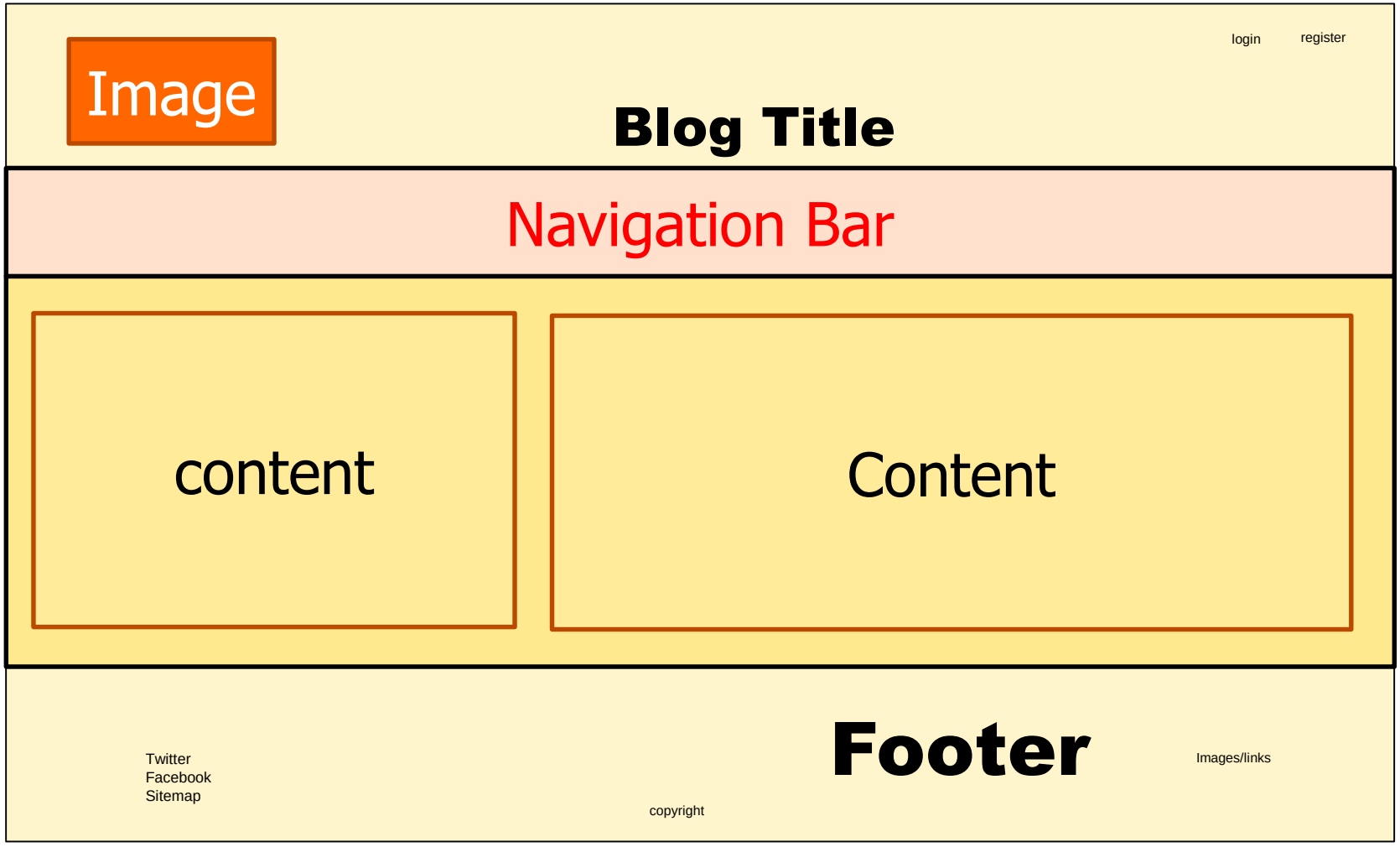
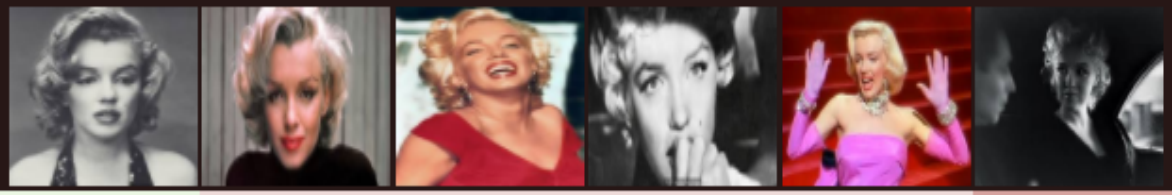


Image
login
register
Blog Title
Navigation Bar
content
Content
Footer
Twitter
Facebook
Sitemap
copyright
Images/links

Marilyn Monroe Tribute



Some Interesting Links

NewBridge Museum Website

Wikipedia page about Marilyn

Article about Marilyn in the Los Angeles Times

Article about Marilyn in the Irish Mirror

Marilyn Monroe

> Imperfection is beauty, madness is genius and it's better to be absolutely ridiculous than absolutely boring. *Marilyn Monroe*

Background

Marilyn Monroe was an American actress, comedienne, singer, and model. She became one of the world's most enduring iconic figures and is remembered both for her winsome embodiment of the Hollywood sex symbol and her tragic personal and professional struggles within the film industry. Her life and death are still the subjects of much controversy and speculation. She was born Norma Jeane Mortenson at the Los Angeles County Hospital on June 1, 1926. Her mother Gladys Pearl Baker was a film-cutter at Consolidated Film Industries. Marilyn's father's identity was never known. Because Gladys was mentally and financially unable to care for young Marilyn, Gladys placed her in the care of a foster family, The Bolenders. Although the Bolender family wanted to adopt Marilyn, Gladys was eventually able to stabilize her lifestyle and took Marilyn back in her care when Marilyn was 7 years old. However, shortly after regaining custody of Marilyn, Gladys had a complete mental breakdown and was diagnosed as a paranoid schizophrenic and was committed to a state mental hospital. Gladys spent the rest of her life going in and out of hospitals and rarely had contact with young Marilyn. Once Marilyn became an adult and celebrated as a film star, she paid a woman by the name of Inez Melson to look in on the institutionalized Gladys and give detailed reports of her progress. Gladys outlived her daughter, dying in 1984.

Her Personal Life

Movie Career

Her first film was in 1947 with a bit part in The Shocking Miss Pilgrim (1947). Her next production was not much better, a bit in the eminently forgettable Scudda Hoo! Scudda Hay! (1948). Two of the three brief scenes in which she appeared wound up on the cutting room floor. Later that same year, she was given a somewhat better role as Evie in Dangerous Years (1947).

However, Fox declined to renew her contract, so she went back to modeling and acting school. Columbia Pictures then picked her up to play Peggy Martin in Ladies of the Chorus (1948), where she sang two numbers. Notices from the critics were favorable for her, if not the film, but Columbia dropped her. Once again Marilyn returned to modeling. In 1949, she appeared in United Artists' Love Happy (1949). It was also that same year she posed nude for the now famous calendar shot which was later to appear in Playboy magazine in 1953 and further boost her career. She would be the first centerfold in that magazine's long and illustrious history.

Black Goose Bakery

The delicious baked goods you love at Black Goose Bistro...now available to go!

Fresh from the Oven

BREADS

Our breads are made daily from highest-quality whole grain flour, water, salt, and yeast or sourdough starter. Simply and naturally, and never any preservatives. Patience is key to achieving the proper level of fermentation and baking each loaf to perfection. Available in whole grain, sourdough, olive loaf, classic rye, and potato-onion.

LEARN MORE ABOUT OUR BAKING PROCESS...

MUFFINS

Every day, we offer a large selection of muffins, including blueberry, multi-berry, bran, corn, lemon-poppyseed, and chocolate. Our muffins are made from scratch each day. Stop by to see ou

LE

Hours

MONDAY: 5am to 3pm

TUESDAY: 5am to 3pm

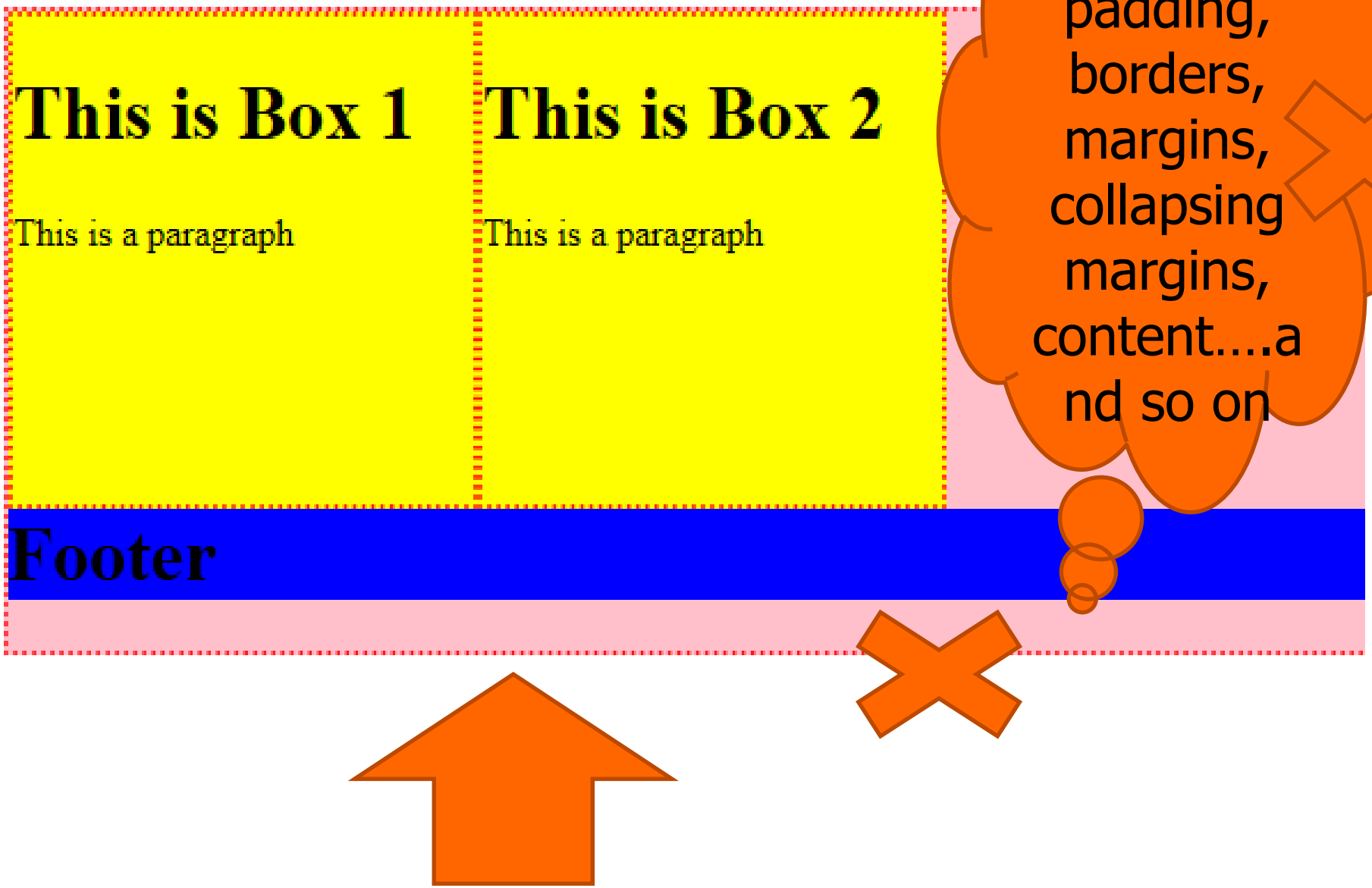
WEDNESDAY: 5am to 3pm

THURSDAY: 5am to 3pm

FRIDAY: 5am to 3pm

SATURDAY: 6am to 4pm

SUNDAY: 6am to 4pm


This is Box 1
This is a paragraph
This is Box 2
This is a paragraph
Footer
Width, height, padding, borders, margins, collapsing margins, content….and so on

A little more symmetrical…

Problems with float layout

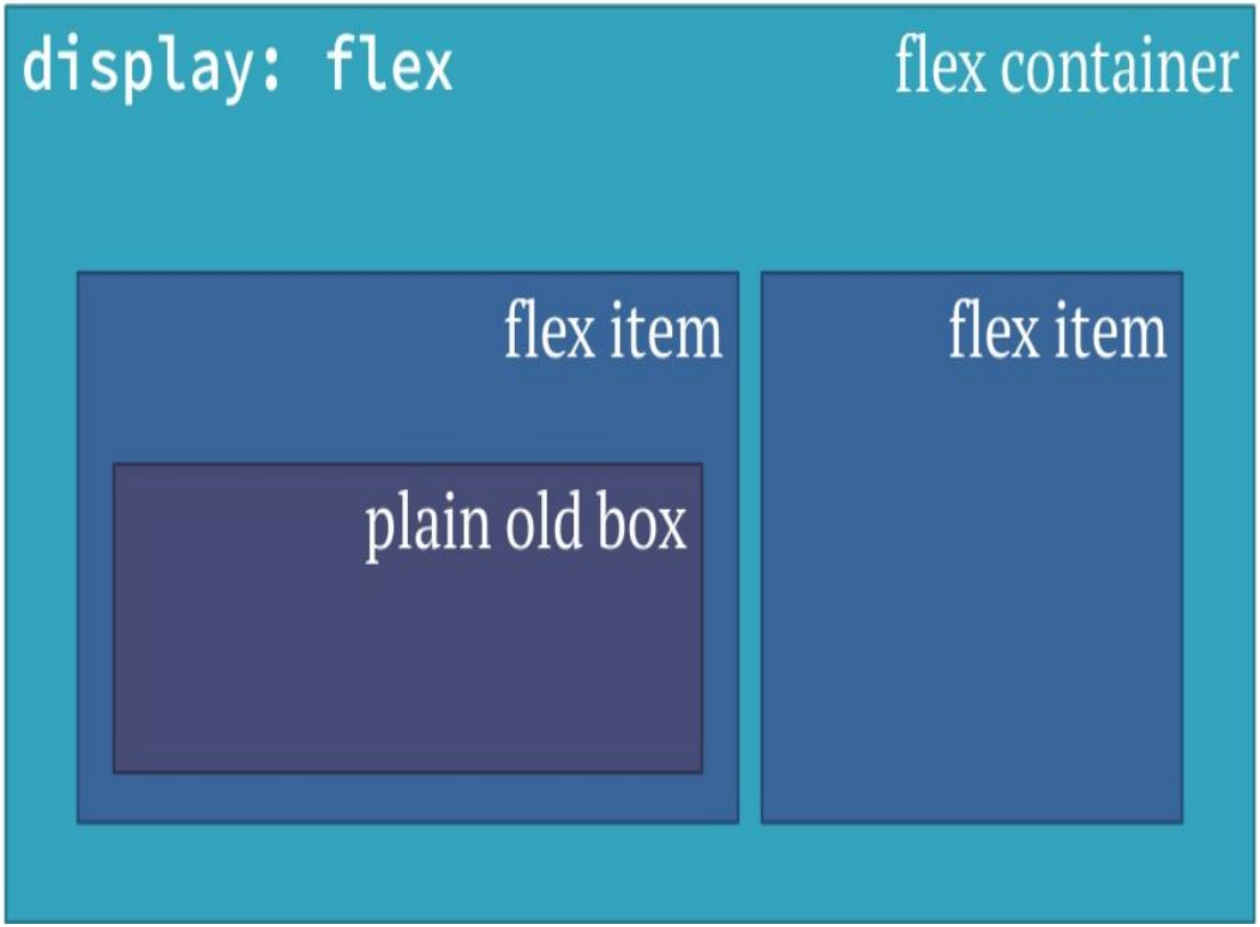
Difficulty with containment

Visual location somewhat tied to HTML order

- No built-in equal-height columns
- No float:center

Clip slide

Turn it on



Flex layout

1. Turn it on display: flex
2. flex item
3. flex container
4. plain old box
5. flex-direction specifies orientation of flex items' layout

row (default) row-reverse
column column-reverse

flex-direction

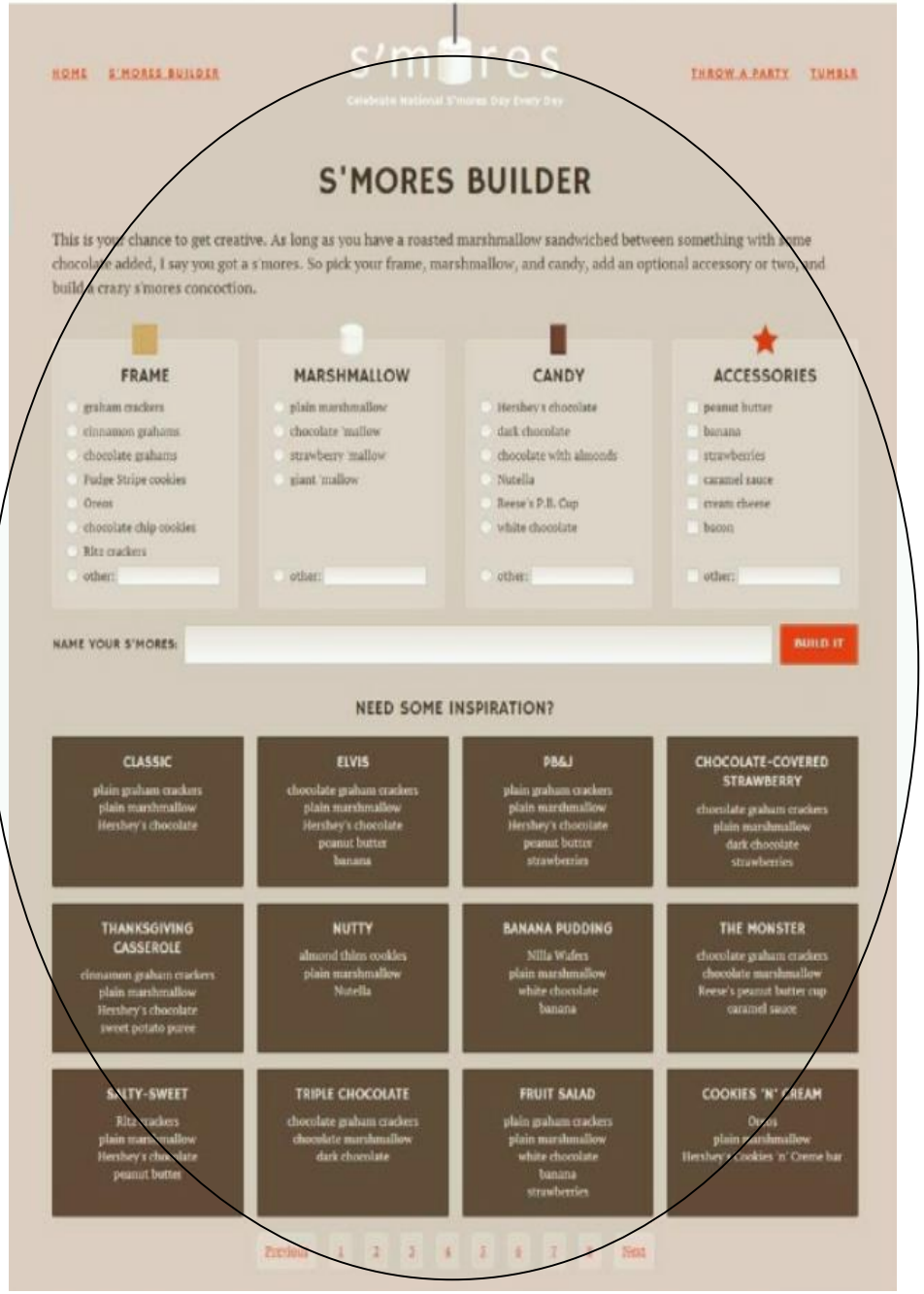
specifies orientation of flex items' layout

`column`

`row`
(default)

`column-reverse`

`row-reverse`



Lets have a look at a flexed website

http://www.smoresday.us/builder.html

Demo: horizontal navigation

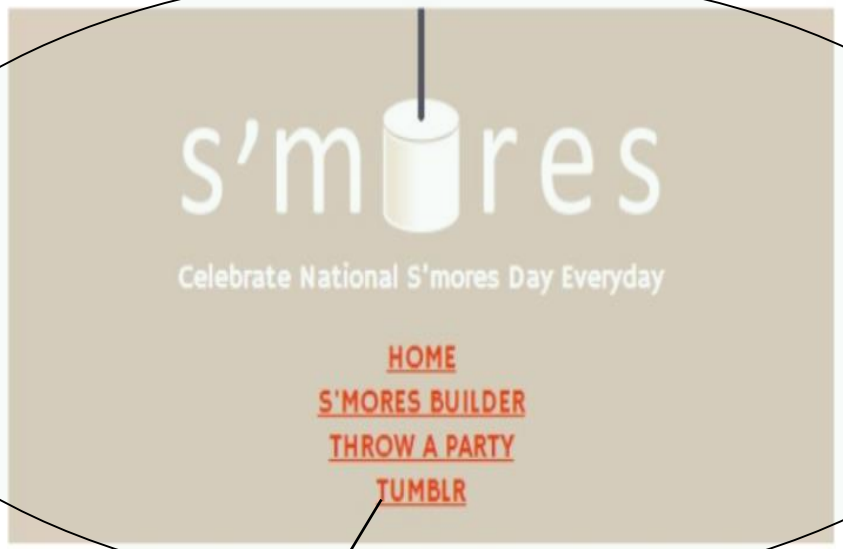
1. Turn <ul> into flex container:

```
.list-nav {
  display: flex;
  flex-direction: row; /* default */
}
```

2. Children <li> become flex items laid out on single horizontal line

Demo: horizontal navigation

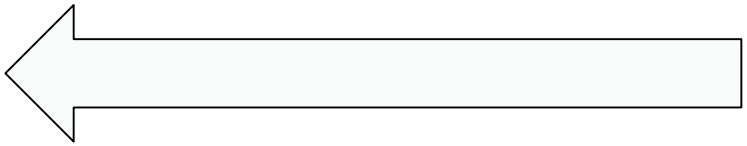
Before



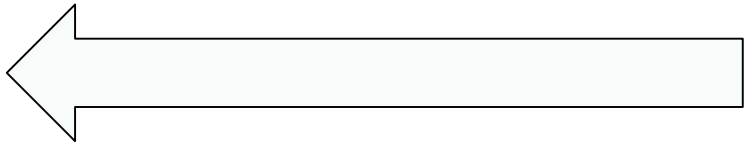
After

Demo: horizontal navigation

Turn <ul> into flex container:



```
.list-nav
{
display: flex;
flex-direction: row;
 }
```



Children <li> become flex items laid out on single horizontal line

Sophisticated page layout control

- Style elements into columns/rows using ‘flexible’ boxes

- We’ll learn to use versions of the following properties:

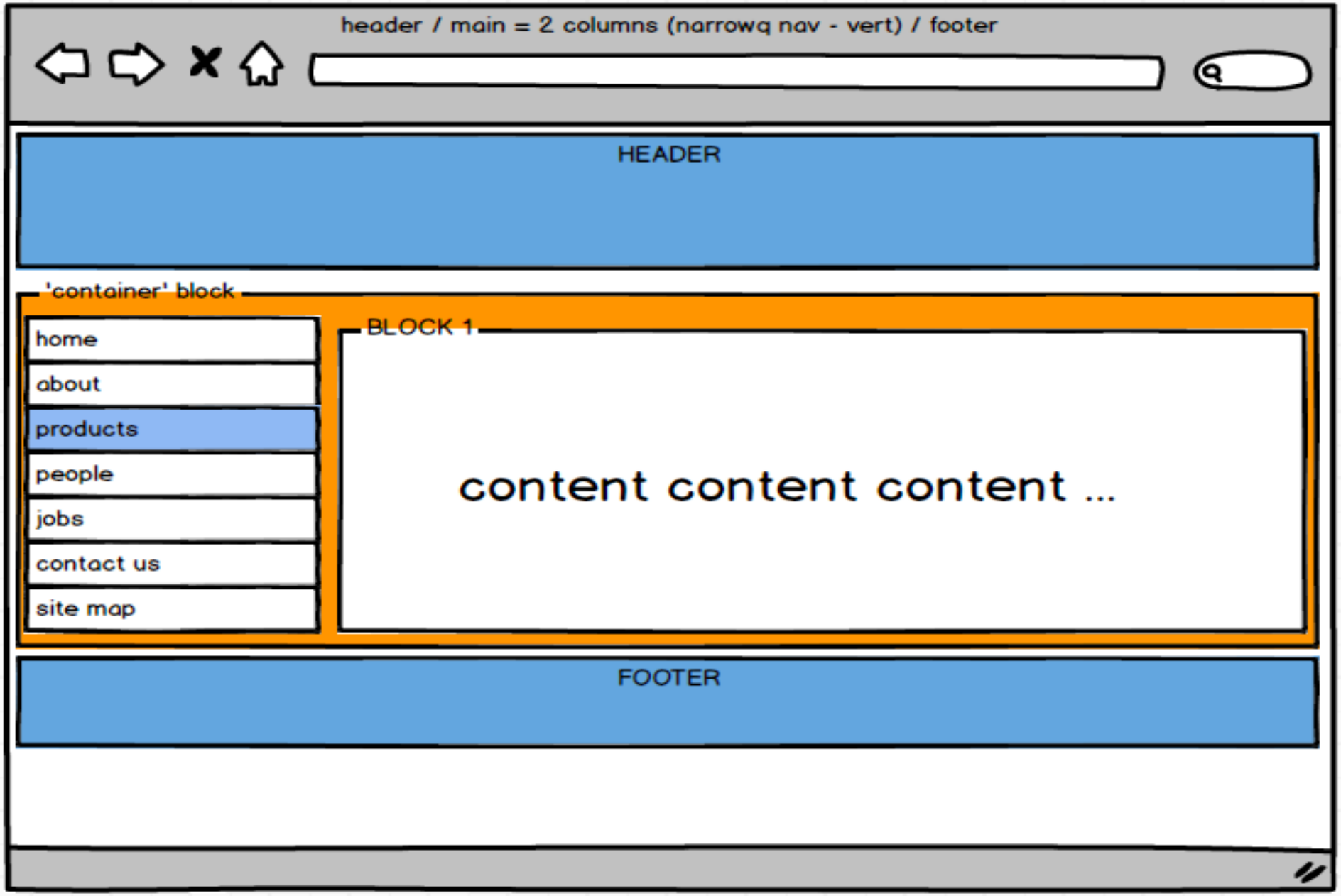
```
display: flex
flex: 1  0   200px
flex-wrap: wrap | nowrap | wrap-reverse
flex-direction: row / column
align-items: center;
```

- For navbars, multi-column layouts, ‘gallery’ pages
  - Straightforward & consistent approach to layouts

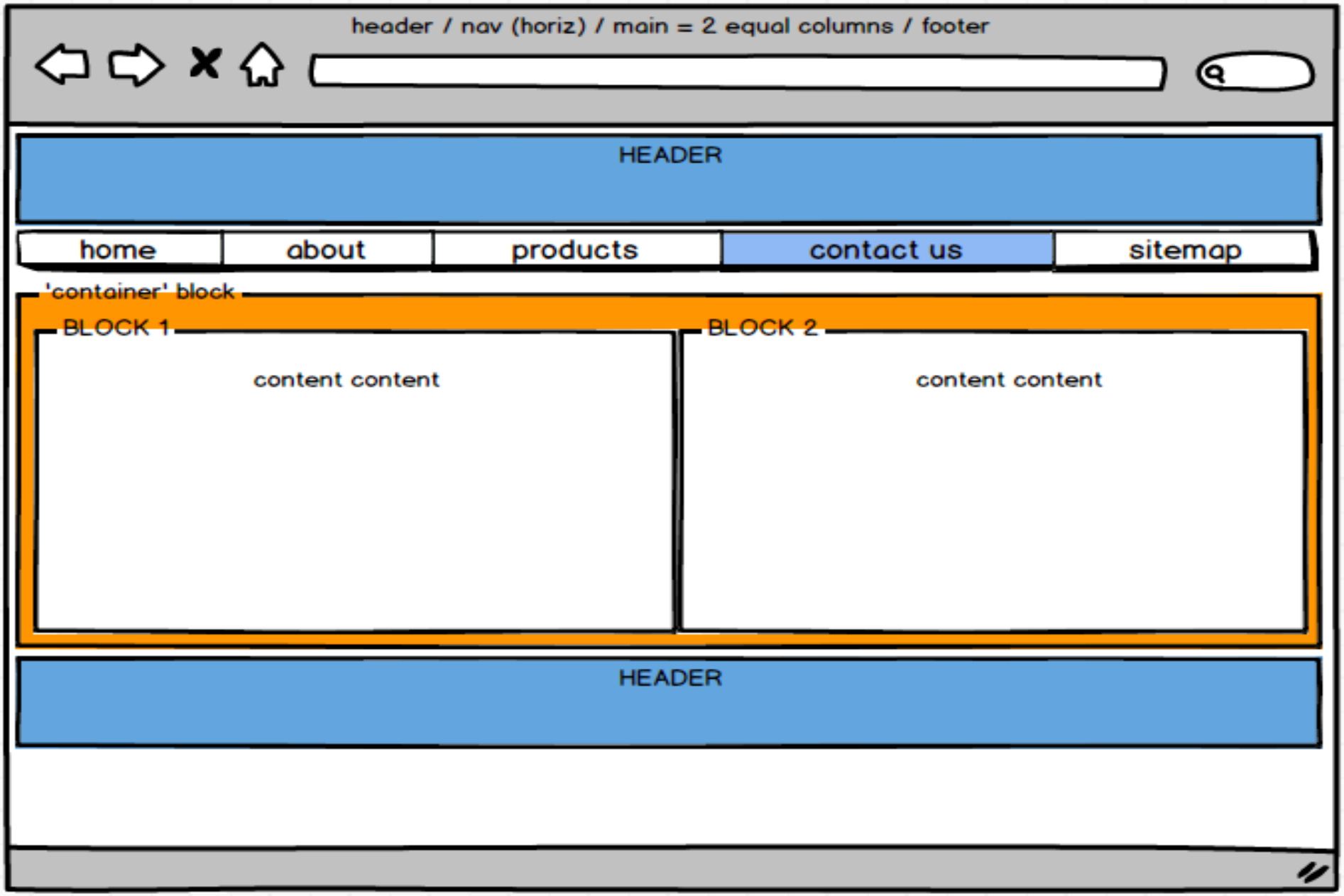
Wouldn't it be nice if ...

- We could keep all the advantages of blocks
  - Margins and padding on all 4 sides
  - They 'fill' available space
- But we control whether blocks line up:
  - Like a row (left to right)
  - Like a column (top to bottom)
  - Or a mixture for complex page layouts ...
- Flexible boxes allow us to CHANGE default layouts to allow control over page layouts ...

Example layout easily achieved with flex boxes



So will this ...


header / nav (horiz) / main = 2 equal columns / footer
HEADER
home
about
products
contact us
sitemap
'container' block
BLOCK 1
content content
BLOCK 2
content content
HEADER

Before we worry about the CODE

Understand the RECTANGLES …

KISS (Keep It Simple Silly …)

- Make use of these DEFAULT layouts
  - Don’t add CSS style or extra HTML DIVs etc. if not needed !
  - So we’ll analyze rows then columns then rows etc.
- Typical page layouts naturally have several self-contained horizontal blocks
  - These fit within default block-level layout
  - And may not need any special layout CSS …

Last updated: Tuesday 5th November 2013

Login | Register

Maynooth Taekwon-Do School

Here are links to some related Taekwon-Do associations

- Irish National Taekwon-Do Association
- All Europe Taekwon-Do Federation
- International Taekwon-Do Federation

Follow us on Facebook

Follow us on Twitter

Divide your page into simple horizontal sections to start with and leave the mre complex layouts for now..

- Each self-contained horizontal section should be a block
  - See how many horizontal lines you can draw
  - That run full width of page …

- Usually we'll have some / all of the following
  - Header
  - Nav
  - Main
  - Footer

Last updated: Tuesday 5th November 2013

Login | Register

header

MAYNOOTH TAEKWON-DO SCHOOL

nav

HOME | ABOUT TAEKWO | NEWS & EVENTS | THEORY & GRADING

INSTRUCTORS | CLASSES | LINKS | CONTACT US

Here are links to some related Taekwon-Do associations

- Irish National Taekwon-Do Association
- International Taekwon-Do Federation

Main – will have 3 column layout

footer

Maynooth Taekwon-Do School © 2010-2013 • Privacy Polic

Website development by Matt Smith , email: webmaster

on Facebook

Follow us on Twitter

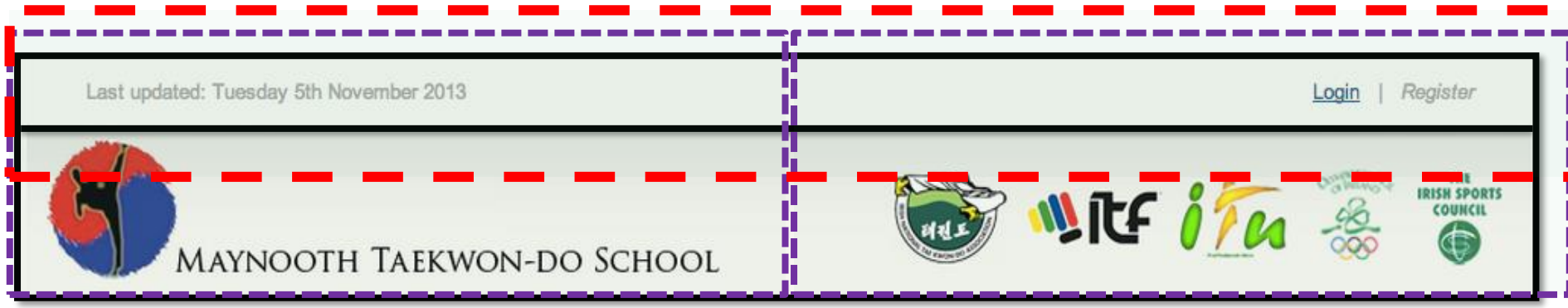
This will help in development and will help reduce the complexity of development

- Limit yourself to 4 or 5 small page sections to code

- We now have identified 4 horizontal blocks:
  - header
  - nav
  - main
  - Footer

- We'll worry about each individually
  - Although sometimes 2 or 3 can be coded together…

Step 2: Analyse blocks on your page in terms of vertical sections or what will be a horizontal section

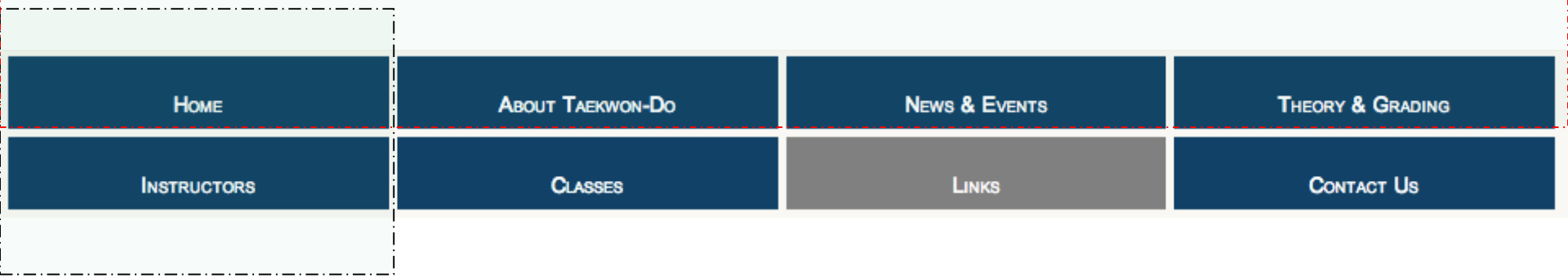


- Options for this layout
  - Have a single 'header' element, with 2 sections
  - Have two 'header' elements, with different IDs

Is there similar layout within each block??

- Yes – both can be treated as split into 2 equal columns
  - So we'll treat these together, since can have same rules for each section of the header

(2b) nav



- Here we see 2 rows of 4 columns
- We could treat this as 2 nav lists
  - Row1: home / about / news / theory
  - Row2: instructors / classes / links / contact
- Or we could treat this as one nav block
  - **with wrapping when elements might be squashed smaller than a quarter width etc.**
- Choose whichever is most straightforward

(2c) main

Here are links to some related Taekwon-Do associations

- Irish National Taekwon-Do Association

- All Europe Taekwon-Do Federation

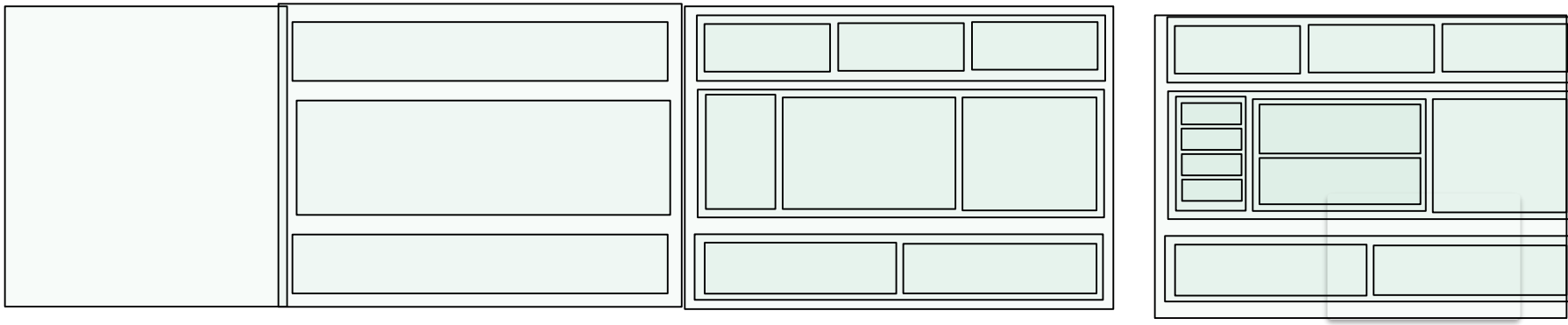
- International Taekwon-Do Federation

- So 2 columns / vertical sections
- Right hand section then split into 3 horizontal sections
  - Each of these split into 2 columns
  - (or perhaps we could use a simple 'float' for these images)

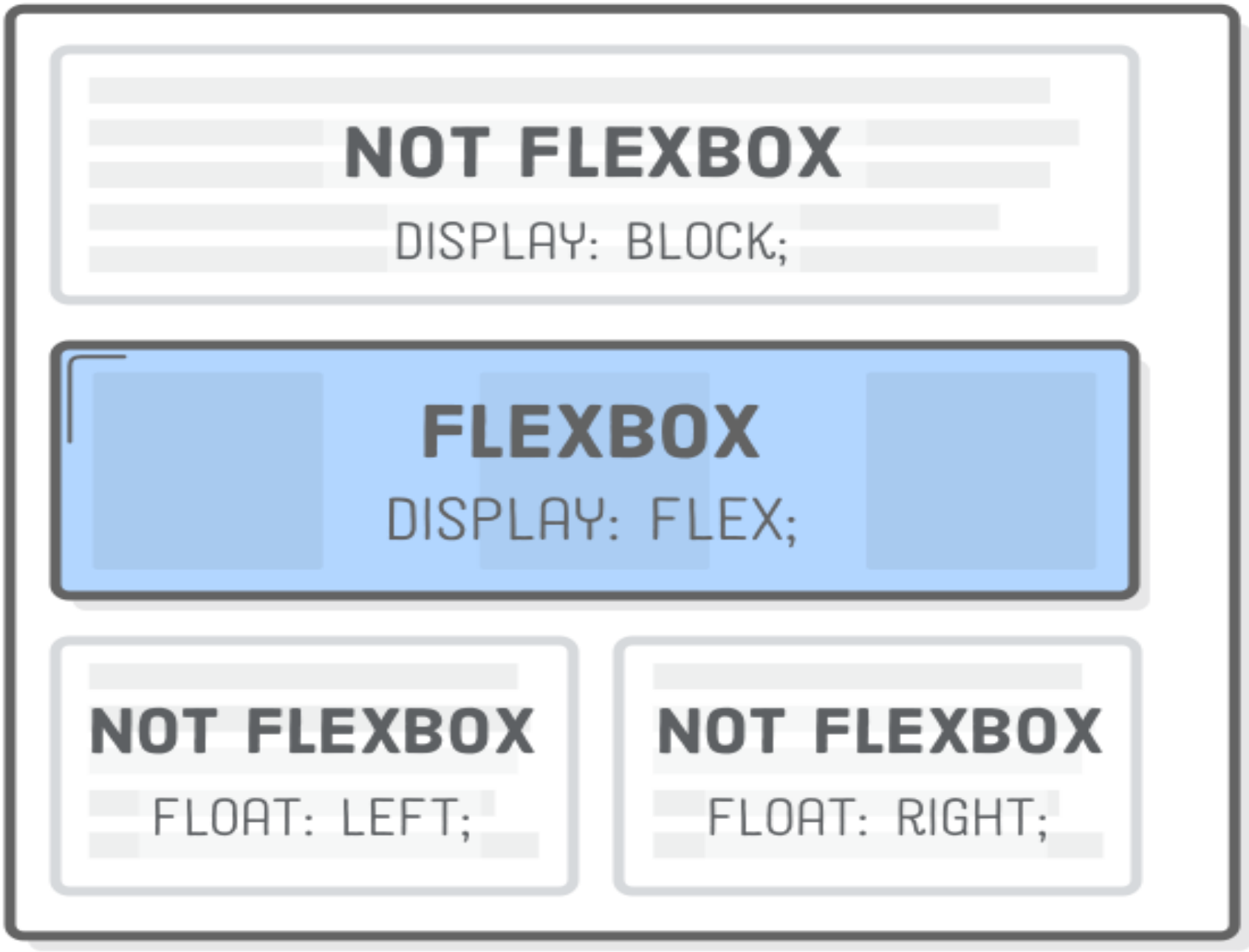
And so on …

- So for each horizontal section
  - If required, identify vertical sections within
  - for each vertical section within, identify horizontal sections
  - And so on …

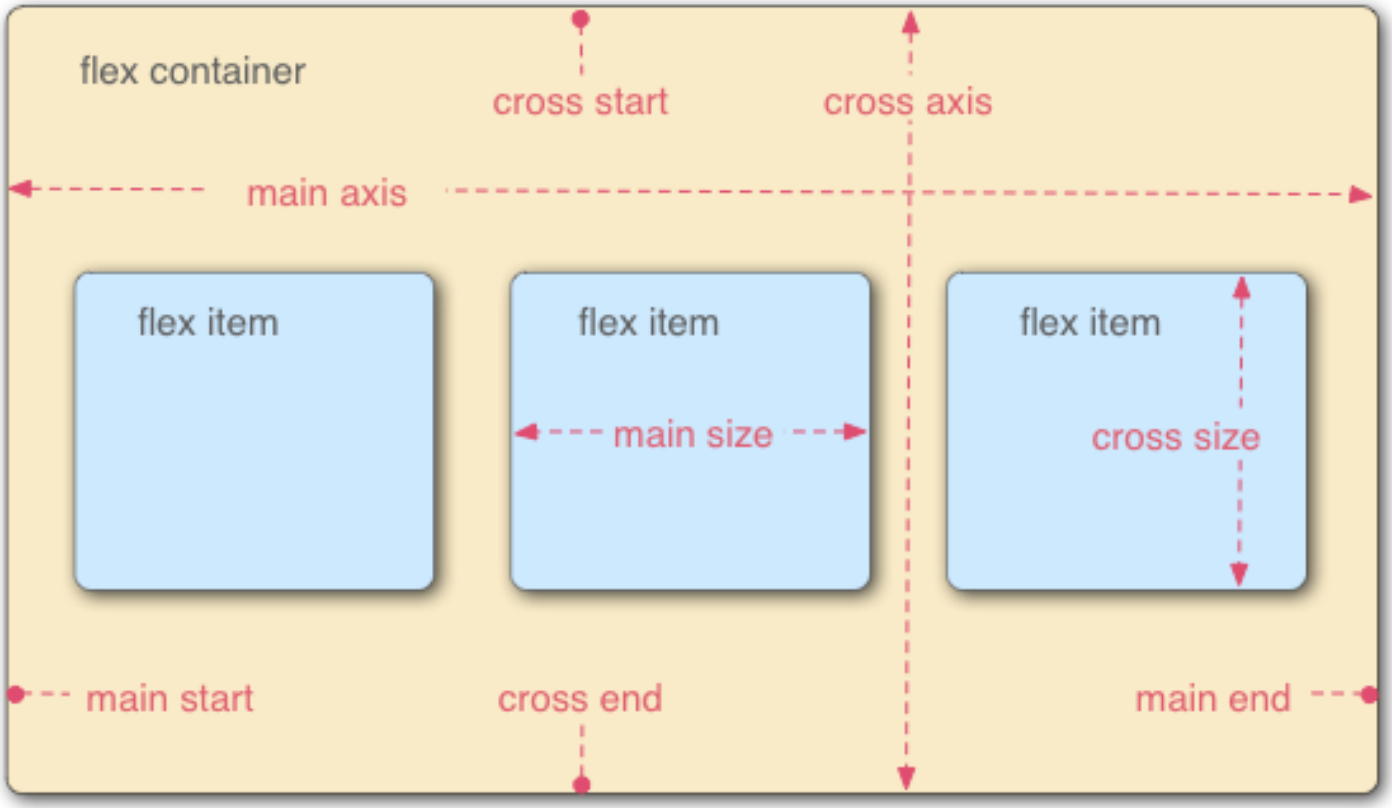
- A ‘recursive’, rectangle sub-division of page structure



blocks_display


NOT FLEXBOX
DISPLAY: BLOCK;
FLEXBOX
DISPLAY: FLEX;
NOT FLEXBOX
FLOAT: LEFT;
NOT FLEXBOX
FLOAT: RIGHT;

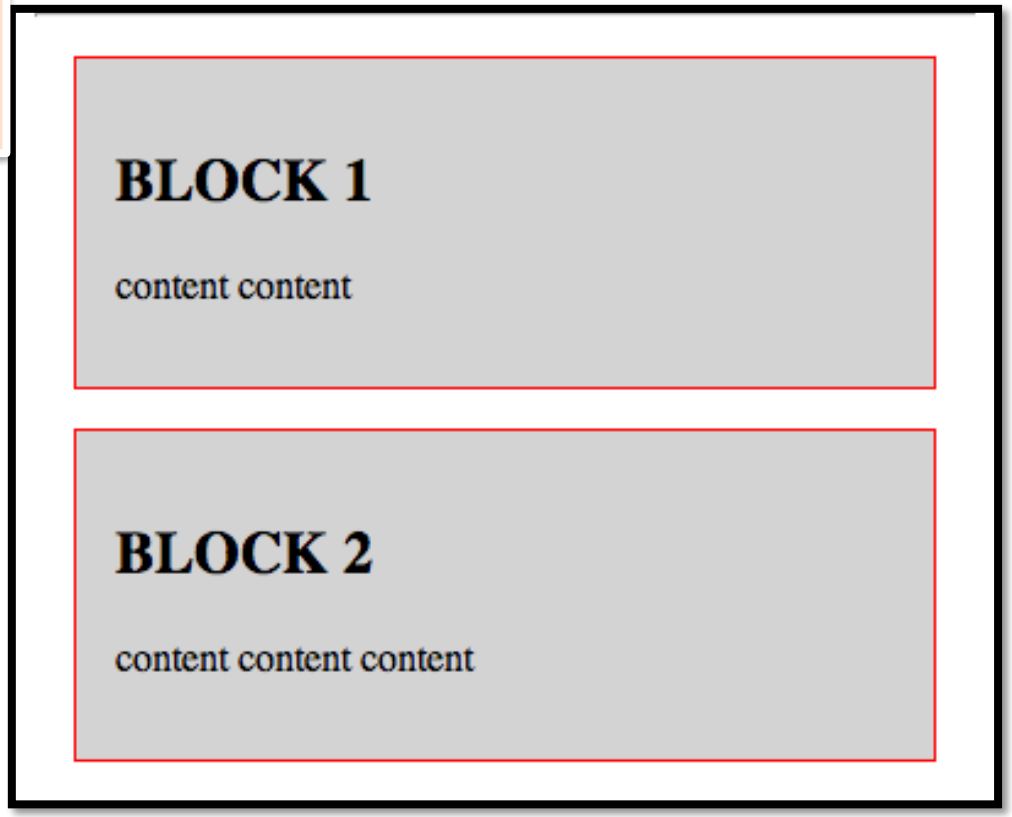

Header
Home
About
Products
Contact
Aside 1
Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.
Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.
Aside 2
Footer



2 important core concepts:

- **Flex ‘container’ (parent block)**
- **Flex ‘item’ (child blocks)**
- **OR use a wrapper…same code different name**
- **e.g **

- Each section has a bit of style (red border etc.)
- So we can see its boundaries



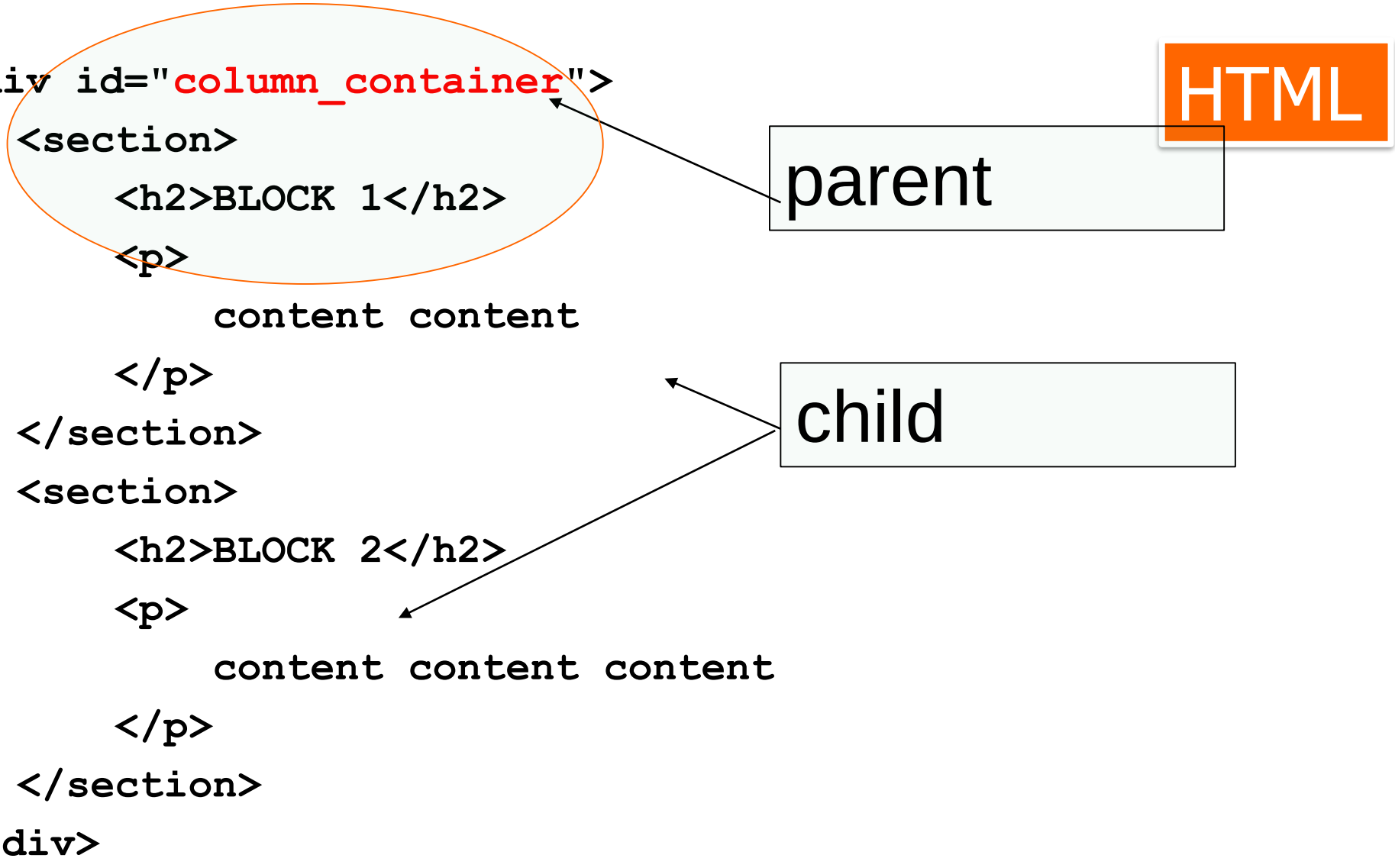

```
<section>
    <h2>BLOCK 1</h2>
    <p>
        content content
    </p>
</section>
<section>
    <h2>BLOCK 2</h2>
    <p>
        content content content
    </p>
</section>
```

HTML

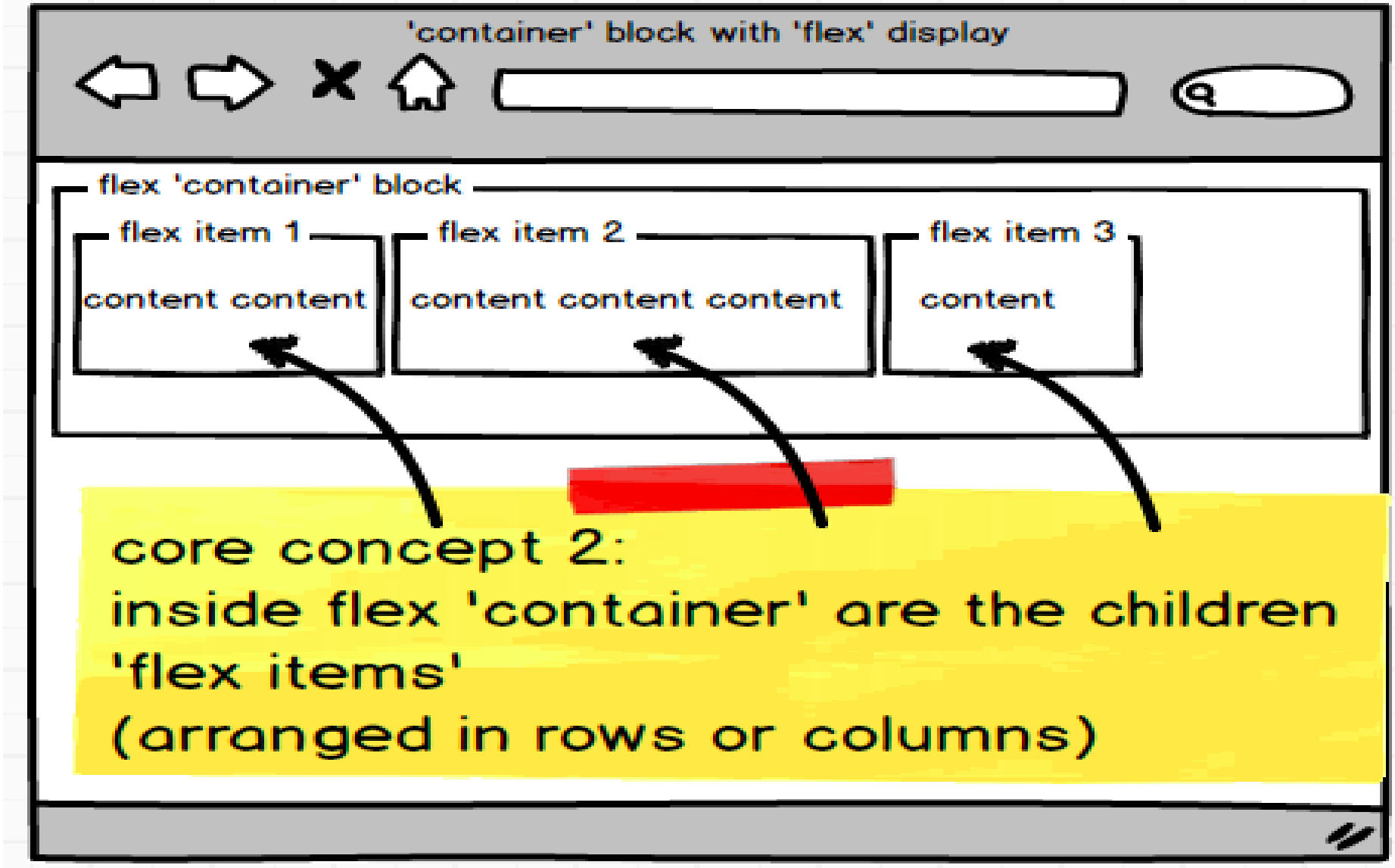
blocks_in_flex_parent

We place the 2 SECTIONS inside a DIV parent given the ID = `column_container`

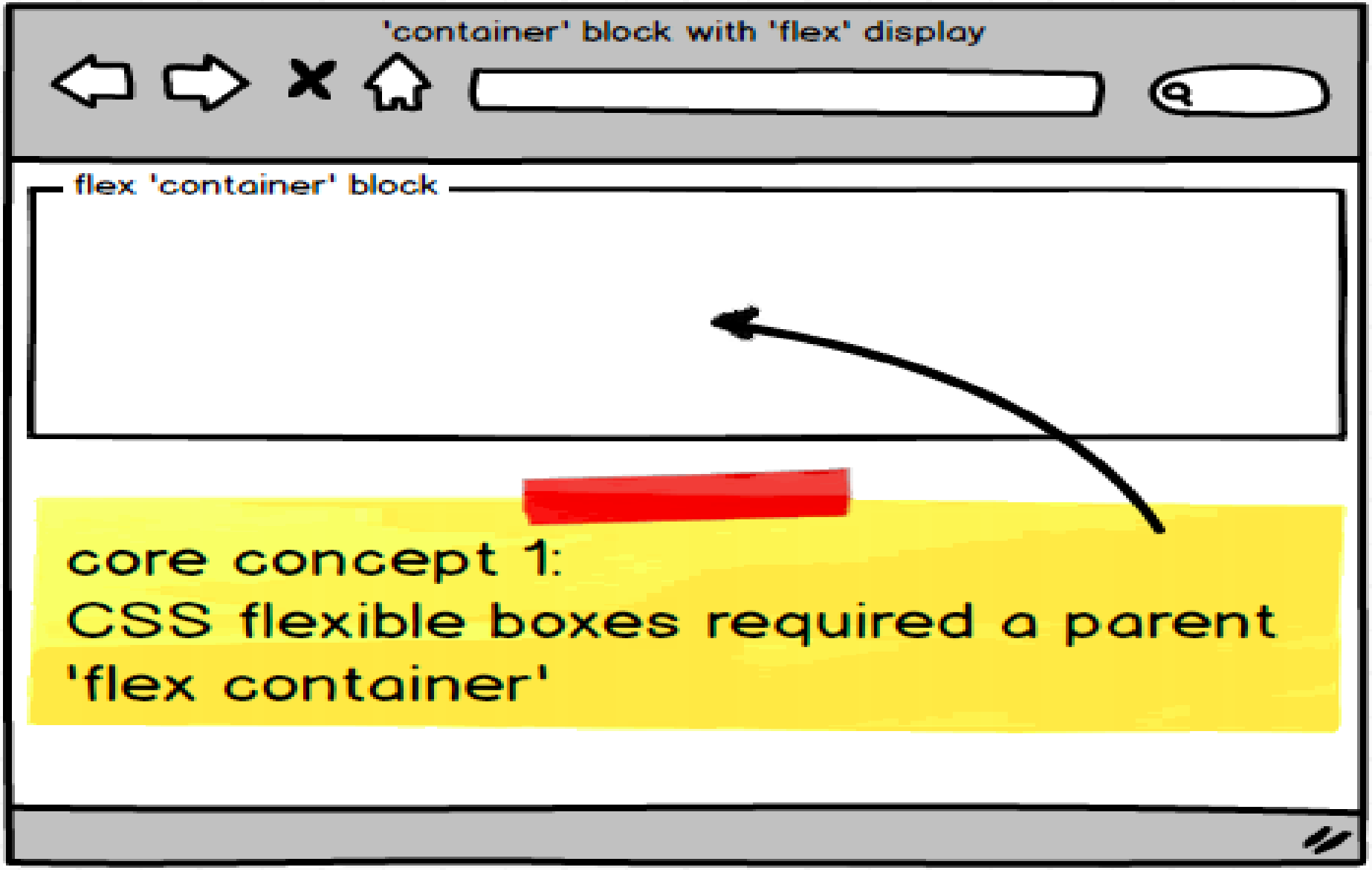
```
<div id="column_container">
   <section>
        <h2>BLOCK 1</h2>
        <p>
             content content
        </p>
   </section>
   <section>
        <h2>BLOCK 2</h2>
        <p>
             content content content
        </p>
   </section>
</div>
```

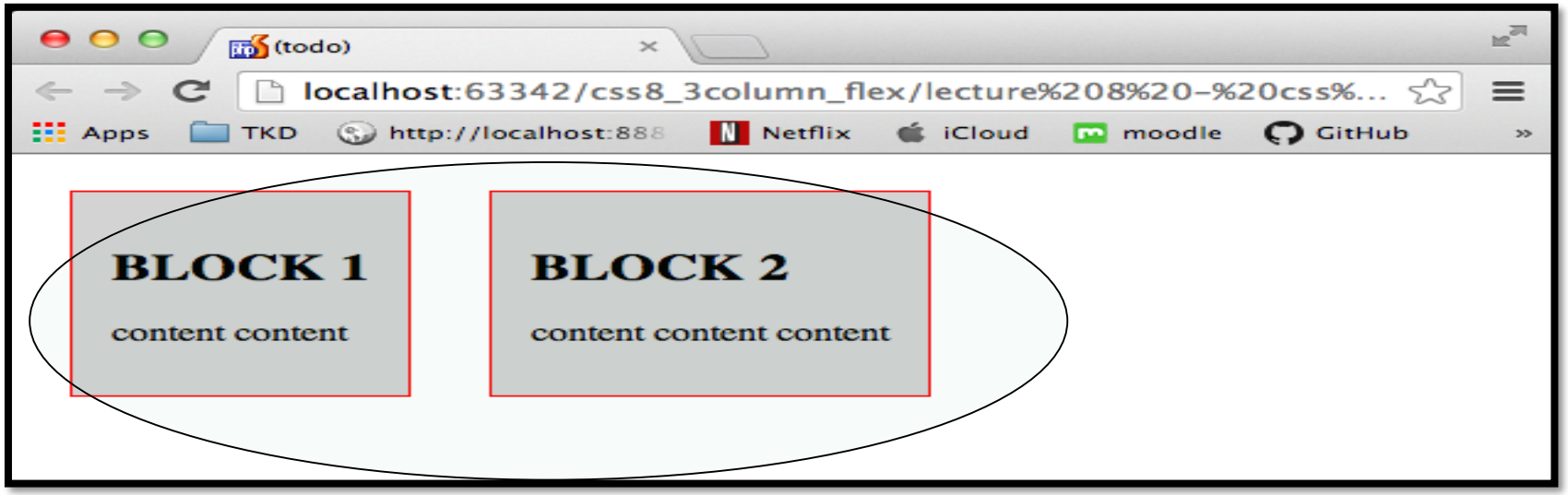


The flex ‘items’
- the children being laid out in the ‘container’



The flex ‘container’ block
whose children can be arranged in rows or columns





- The DIV has been styled as a 'flex container'

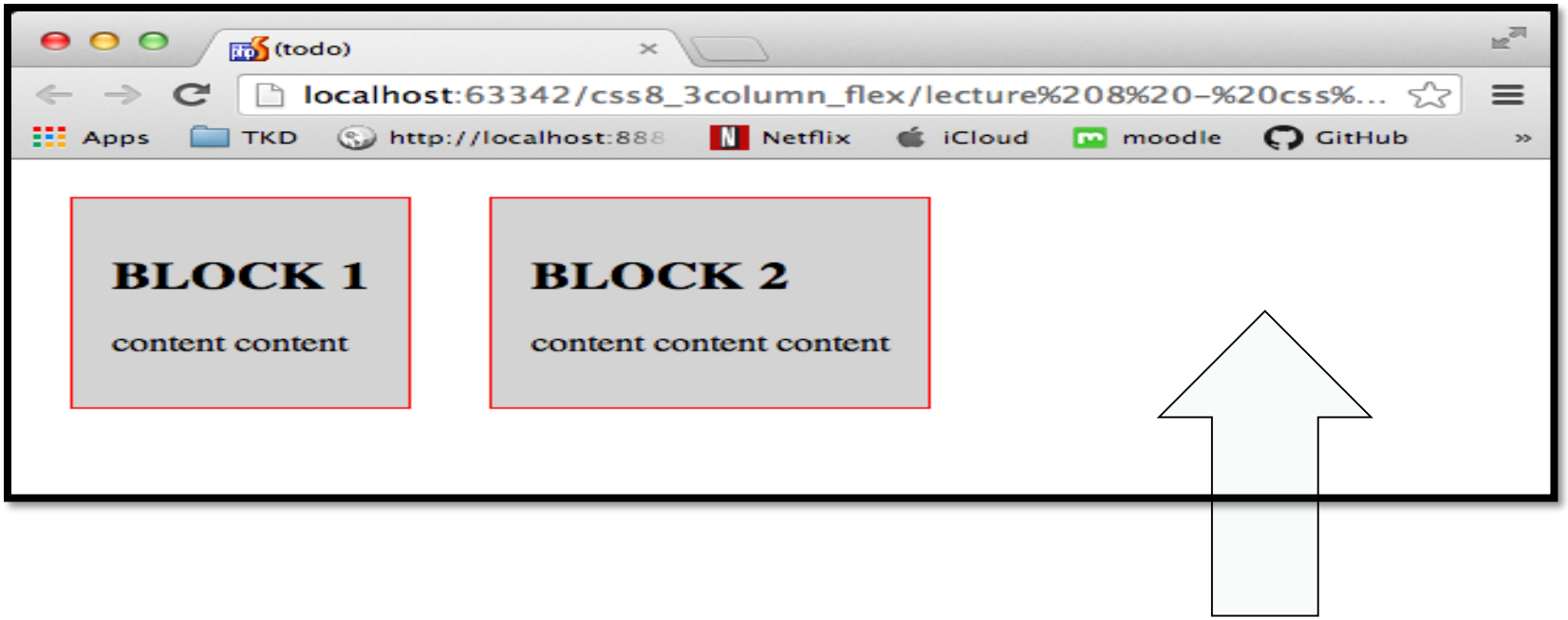
```
display: flex;
```

  - So the 2 SECTION children inside this DIV line up left-to-right (row layout is default for flex containers)

```
#column_container {
    -webkit-display: flex;
    display: flex;
}
```

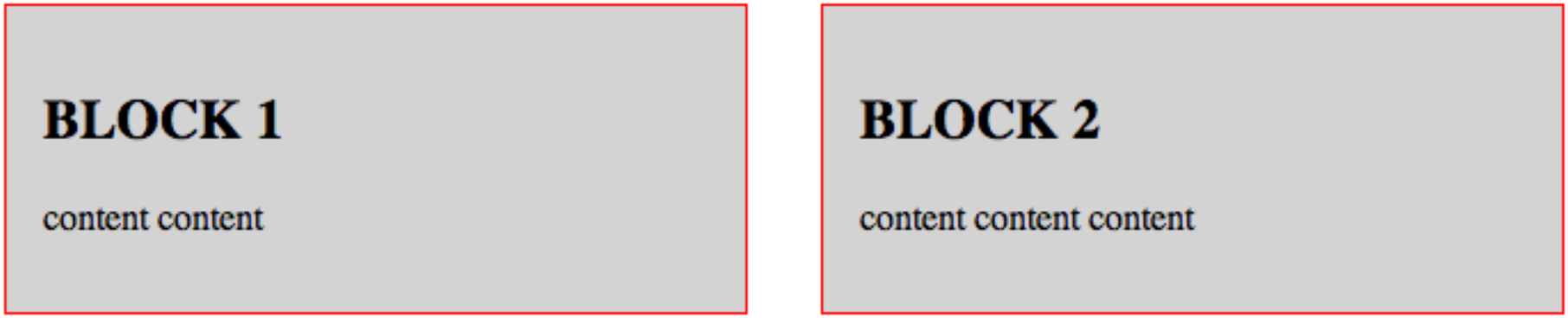
We have flexed the parent

CSS



- Note
  - There is spare space on the right
  - We can tell the children to 'stretch' to use up the spare space …

blocks_stretch_no_wrap



- The HTML is unchanged
- Each section has been styled to equally stretch with any spare width by adding the following code:

```
flex: 1;
```

  - So the 2 SECTION children expand to be 2 equal columns

```
#column_container section {
        flex: 1;
    -webkit-flex: 1;
}
```

CSS

Here is same – but no margins on body or the sections – clean and simple 2 columns …

Column 1

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Haec et tu ita posuisti, et verba vestra sunt. Quae in controversiam veniunt, de iis, si placet, disseramus. Nam de isto magna dissensio est. Ita nemo beato beatior. Duo Reges: constructio interrete.

Column 2

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Haec et tu ita posuisti, et verba vestra sunt. Quae in controversiam veniunt, de iis, si placet, disseramus. Nam de isto magna dissensio est. Ita nemo beato beatior. Duo Reges: constructio interrete.

home | about | sitemap | Contact | Characters

Kirk
Spock
Bones

James T. Kirk

James T. Kirk

The captain of the starship USS Enterprise ...

Friend of Spock and Bones.

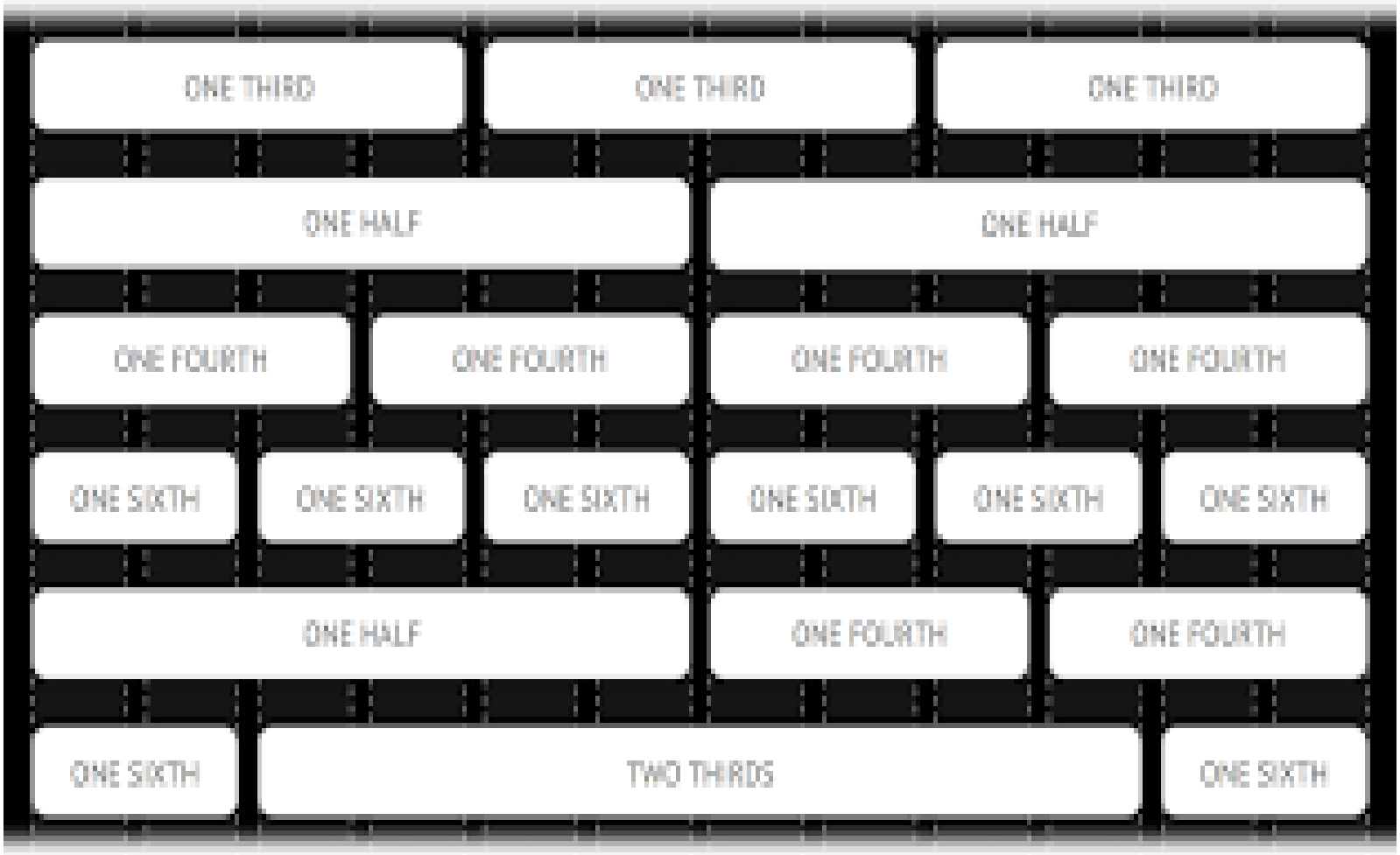
I am para 1

CSS is great!

Copyright © 2014

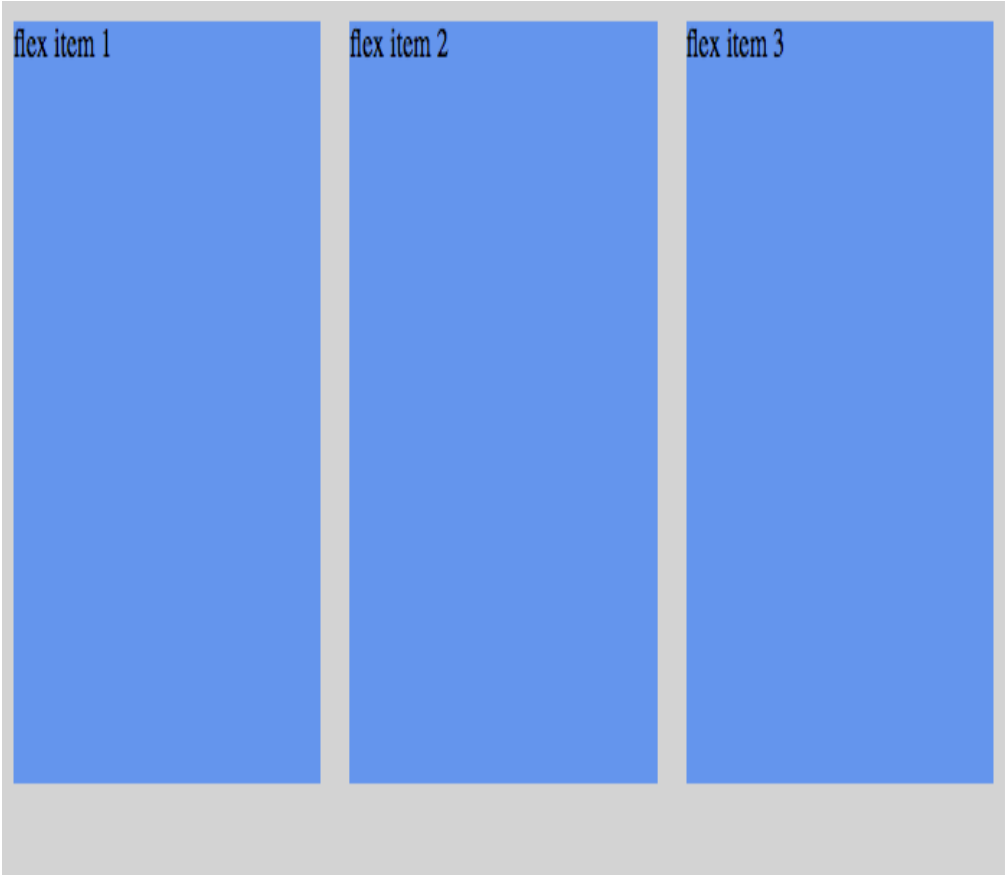
A sample flexed website

Flex Box division Options



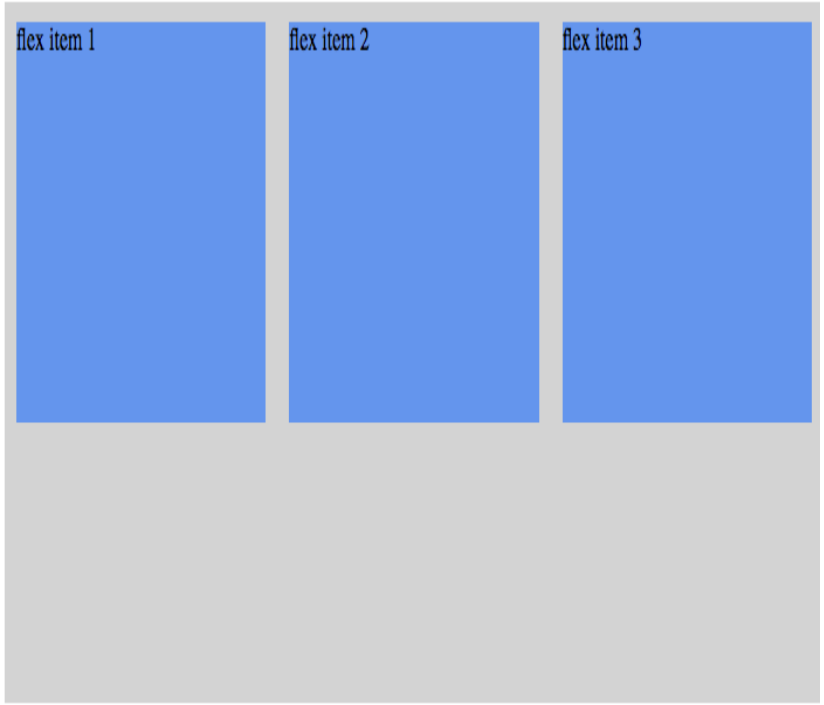
Html and CSS- container flexed with 3 flex boxes within it.

```
<!DOCTYPE html>
<html>
<head>
<style>
.flex-container {
    display: -webkit-flex;
    display: flex;
    width: 700px;
    height: 350px;
    background-color: lightgrey;
}

.flex-item {
    background-color: cornflowerblue;
    width: 500px;
    height: 300px;
    margin: 10px;
}
</style>
</head>
<body>

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



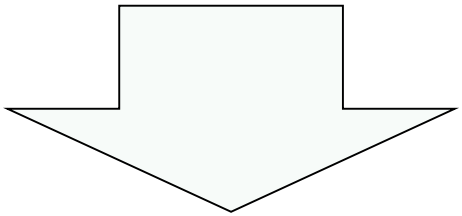
Re-sizing the flexboxes within the container

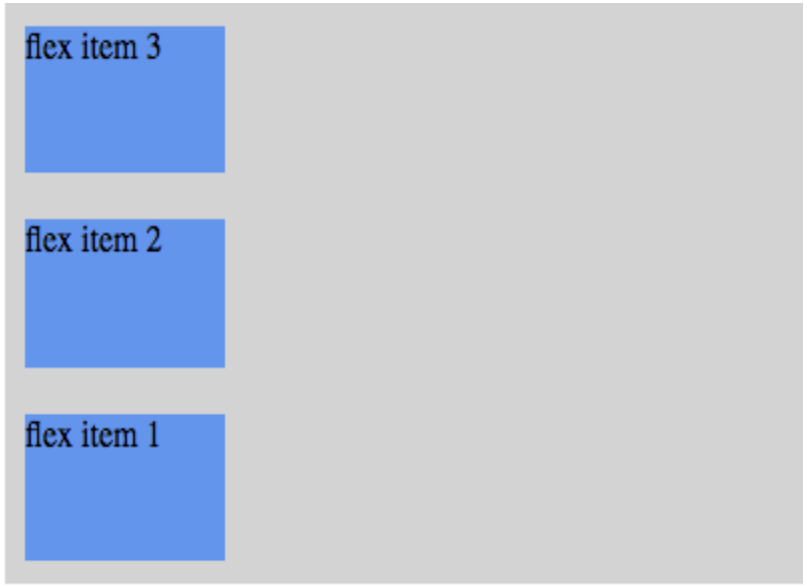
```
.flex-item {
    background-color: cornflowerblue;
    width: 500px;
    height: 200px;
    margin: 10px;
```



Changing the order of the boxes in the container

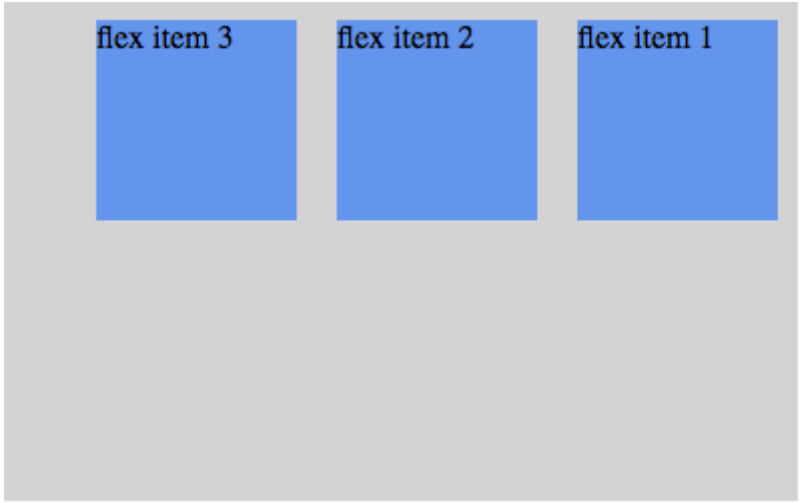
```
<!DOCTYPE html>
<html>
<head>
<style>
.flex-container {
    display: -webkit-flex;
    display: flex;
    -webkit-flex-direction: row-reverse;
    flex-direction: row-reverse;
    width: 400px;
    height: 250px;
    background-color: lightgrey;
}

.flex-item {
    background-color: cornflowerblue;
    width: 100px;
    height: 100px;
    margin: 10px;
}
</style>
</head>
<body>

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



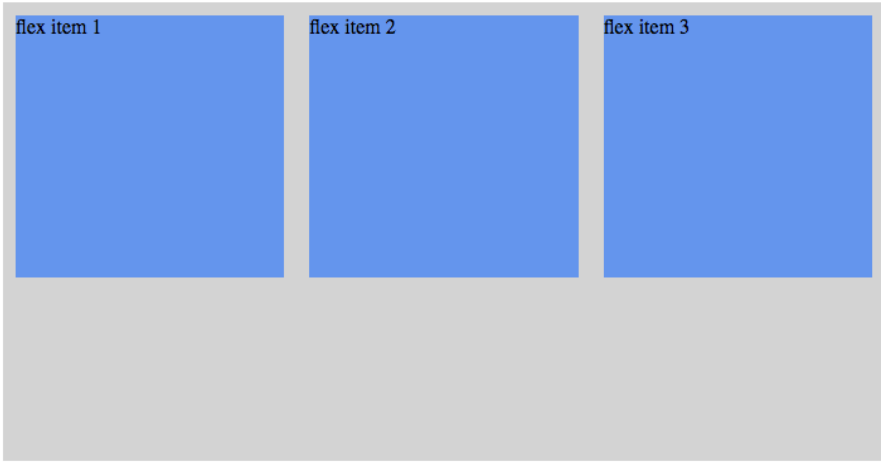


Changing the order of the boxes in the container

```
<!DOCTYPE html>
<html>
<head>
<style>
.flex-container {
    display: -webkit-flex;
    display: flex;
    -webkit-flex-direction: column-reverse;
    flex-direction: column-reverse;
    width: 400px;
    height: 250px;
    background-color: lightgrey;
}

.flex-item {
    background-color: cornflowerblue;
    width: 100px;
    height: 100px;
    margin: 10px;
}
</style>
</head>
<body>

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

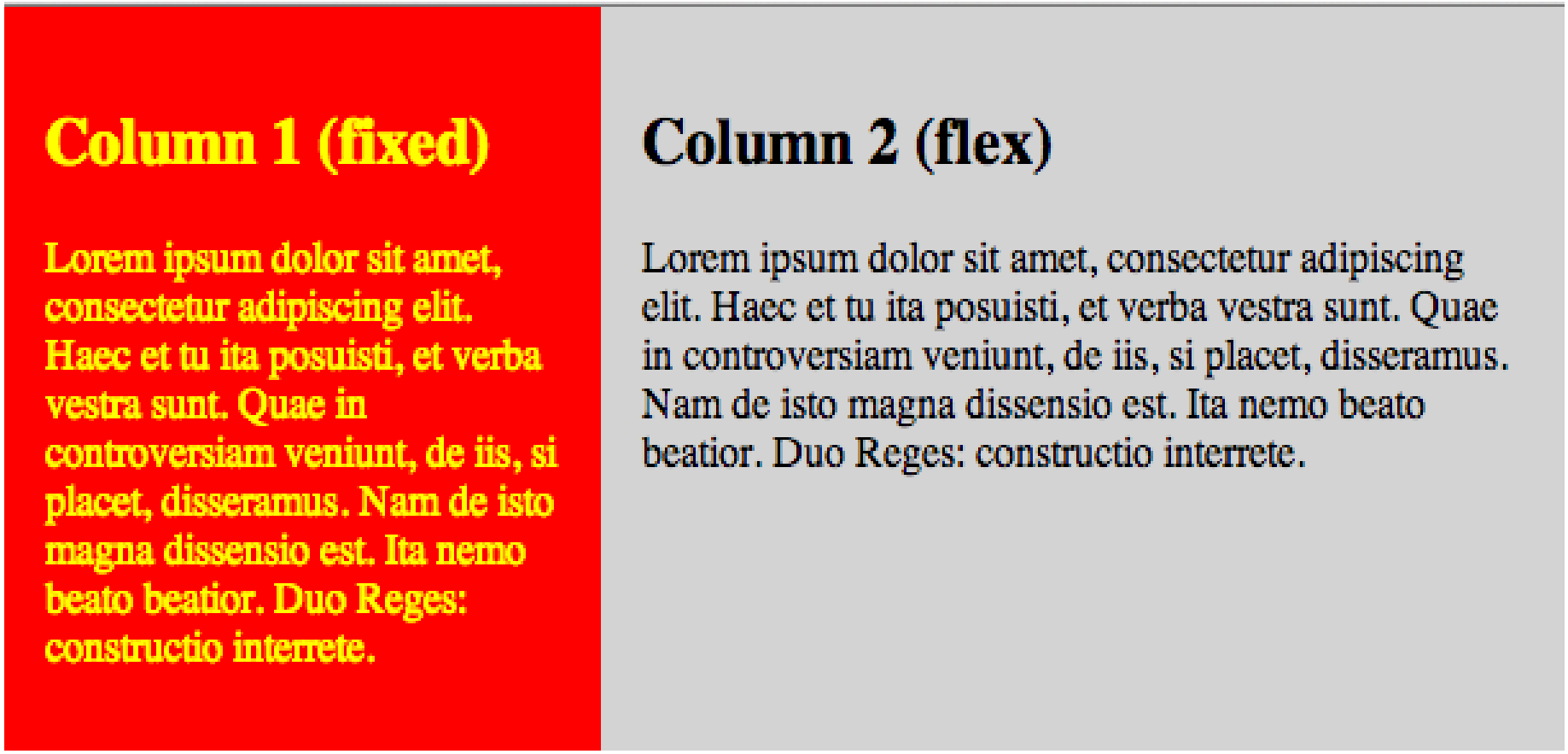


```
<!DOCTYPE html>
<html>
<head>
<style>
.flex-container {
    display: -webkit-flex;
    display: flex;
    -webkit-flex-wrap: nowrap;
    flex-wrap: nowrap;
    width: 300px;
    height: 250px;
    background-color: lightgrey;
}

.flex-item {
    background-color: cornflowerblue;
    width: 100px;
    height: 100px;
    margin: 10px;
}
</style>
</head>
<body>

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



one_fixed_one_flex

Left (red) column is FIXED width (width: 200px)
Right (grey) column flexes (flex: 1)



Each section given a class – so can be styled differently

```
<div id="column_container">
    <section class= "fixed">
        <h2>Column 1 (fixed)</h2>
        <p>
            content content
        </p>
    </section>

    <section class= "flex">
        <h2>Column 2 (flex)</h2>
        <p>
            content content content
        </p>
   </section>
</div>
```

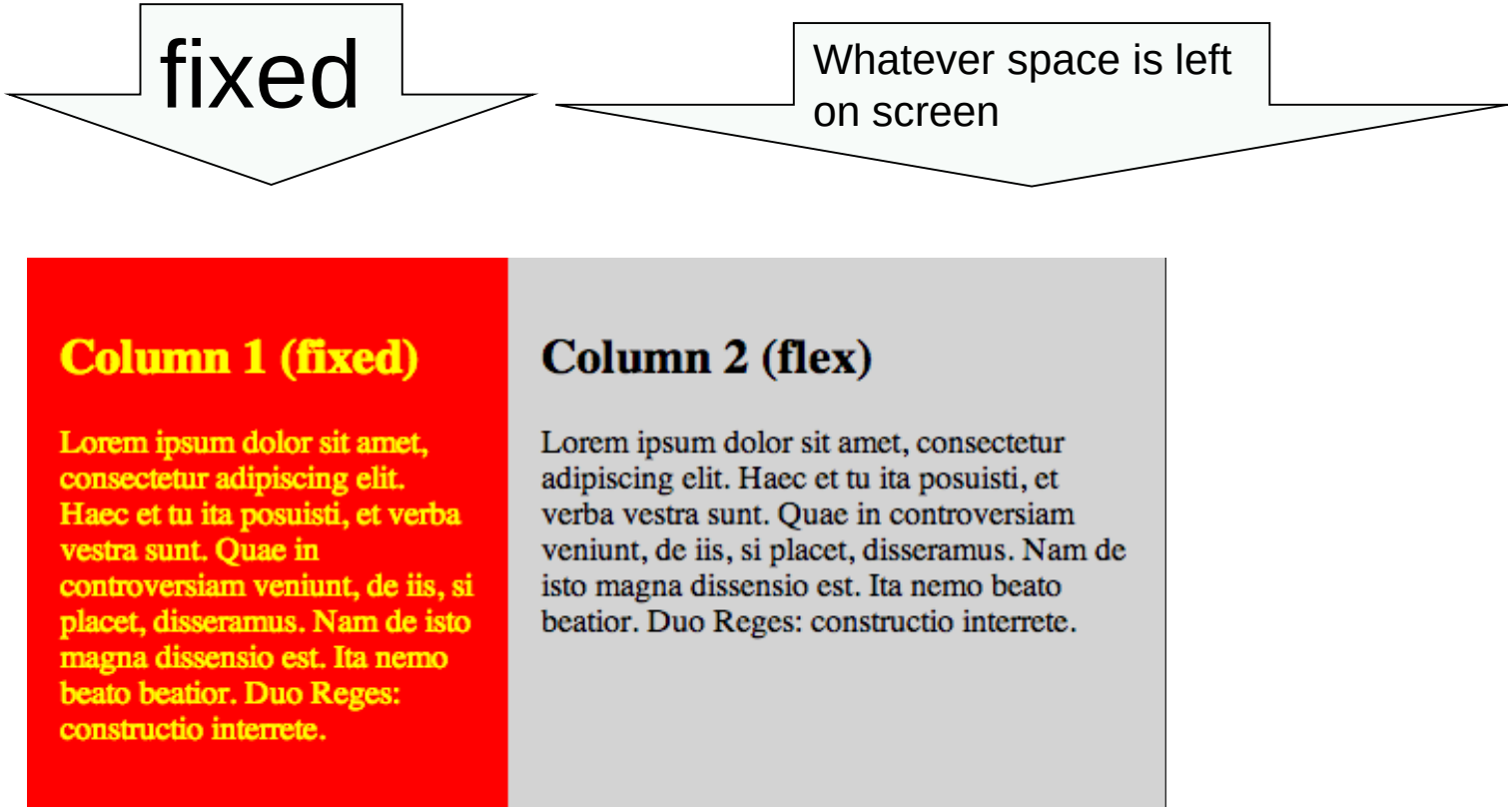
HTML

Only one section class (right column) flexes (so it takes up all the spare width ...)

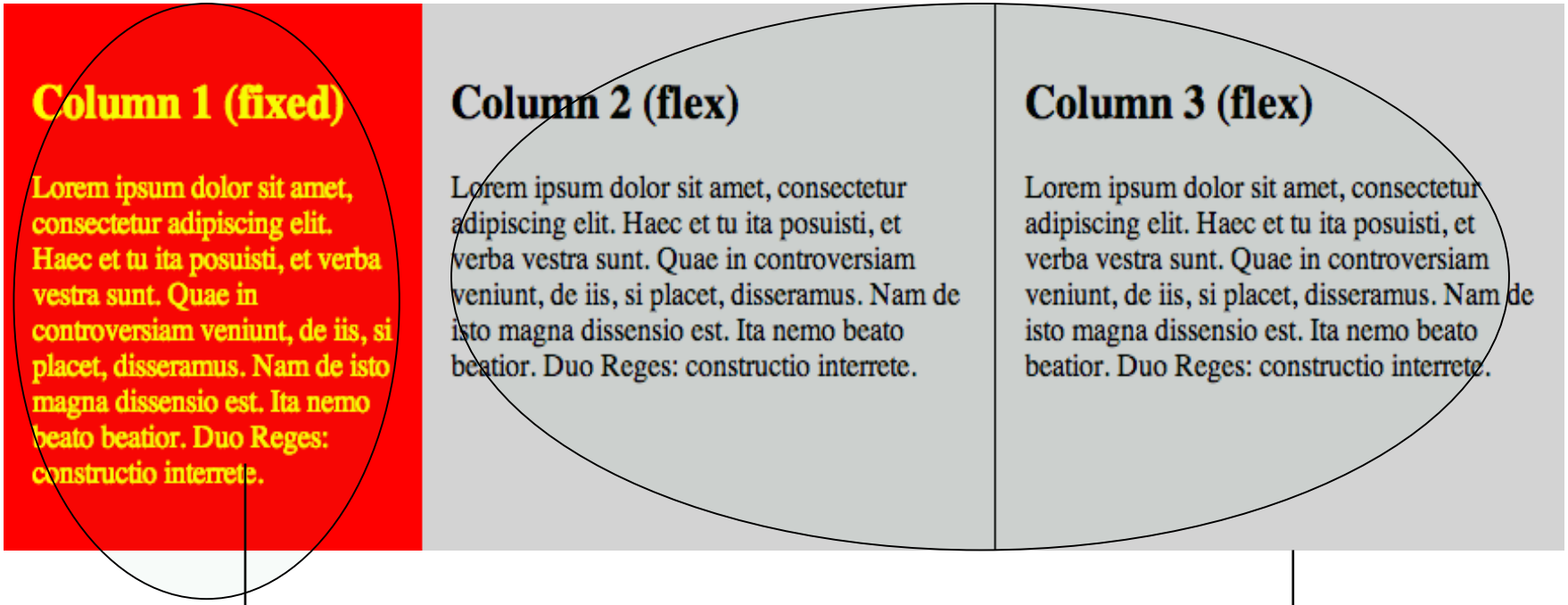
```
#column_container section.fixed {
    width: 200px;
    background-color: red;
    color: yellow;
    padding: 1em;
}

#column_container section.flex {
    -webkit-flex: 1;
    flex: 1;
    background-color: lightgray;
    padding: 1em;
}
```

CSS

Only one section class (right column) flexes (so it takes up all the spare width …)



Add another flex column – no change to CSS



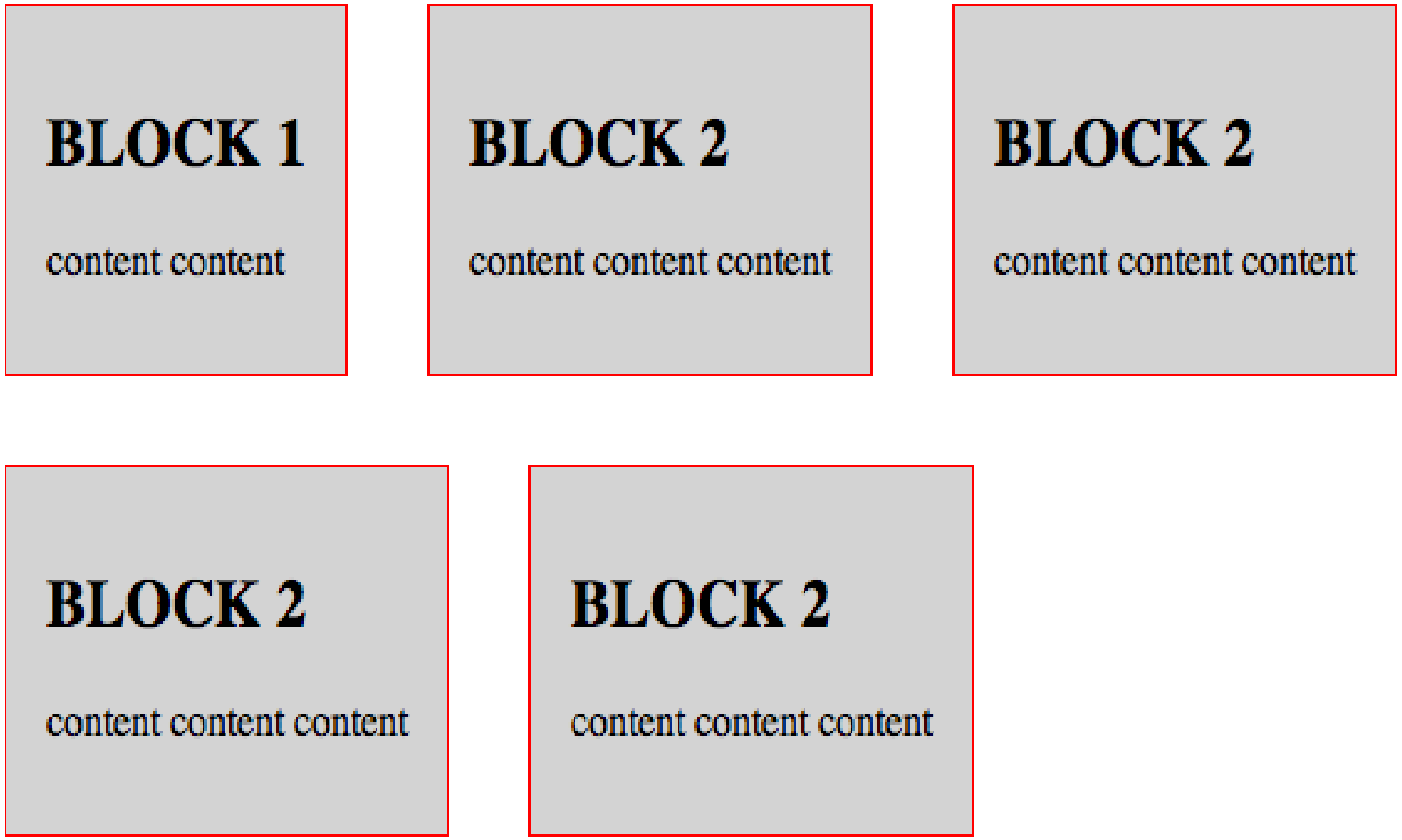

```
#column_container section.fixed {
    width: 200px;
    background-color: red;
    color: yellow;
    padding: 1em;
}

#column_container section.flex {
    -webkit-flex: 1;
    flex: 1;
    background-color: lightgray;
    padding: 1em;
}
```

Just add another section!!

```
<div id="column_container">
     <section class= "fixed">
        <h2>Column 1 (fixed)</h2>
        <p>
            content content
        </p>
     </section>
     <section class= "flex">
         <h2>Column 2 (flex)</h2>
           <p>
            content content content
        </p>
    </section>
     <section class= "flex">
         <h2>Column 3 (flex)</h2>
           <p>
            content content content
        </p>
    </section>
</div>
```

blocks_in_flex_parent_allowing_wrap

Style flex container (parent) to allowing wrapping



Required: style flex container to allow wrapping

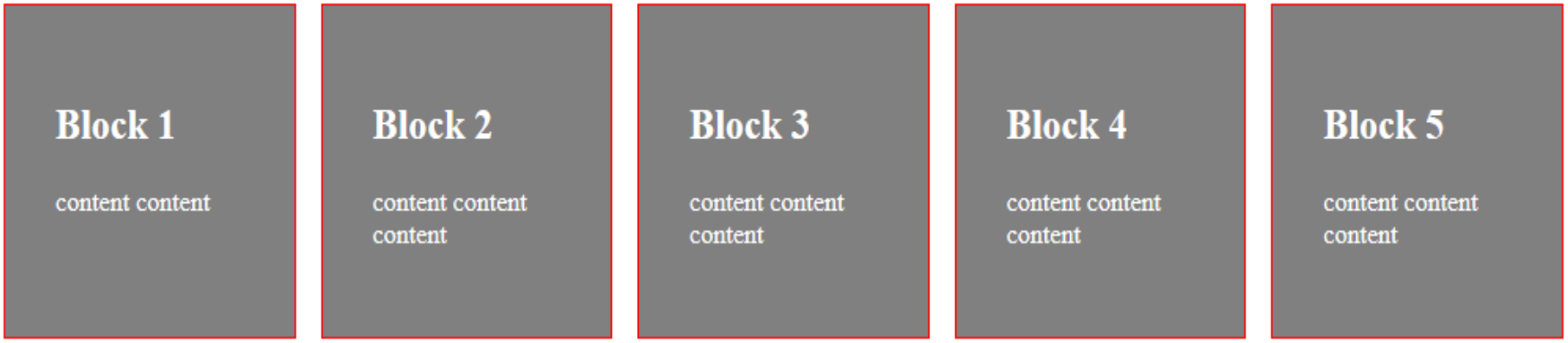
- Add code to flex container (parent) block, to allow wrapping

```
#column_container {
    -webkit-display: flex;
    display: flex;
    -webkit-flex-wrap: wrap;
    flex-wrap: wrap;
}
```

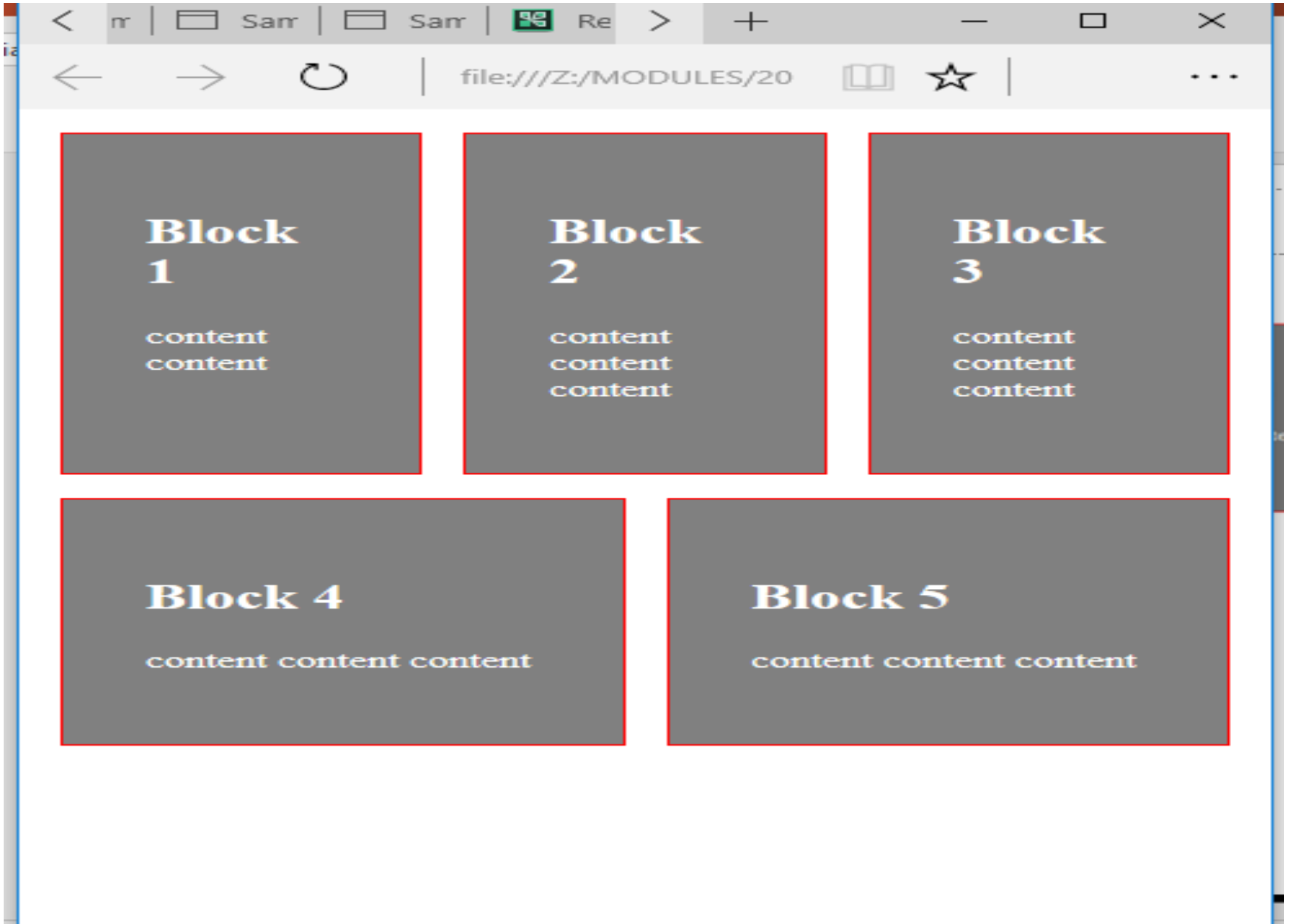
CSS for the sections

```
#column_container section {
        flex: 1 0 200px;
    -webkit-flex: 1;
    background-color: gray;
    color:white;
    border: 1px solid red;
    margin:2em;
Padding:2em;
}
```

Browser window

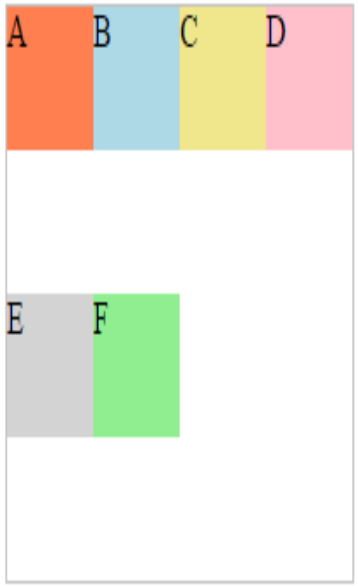


Re-sized browser window



Make the flexible items wrap if necessary:

```
#main {
    width: 200px;
    height: 200px;
    border: 1px solid #c3c3c3;
    display: -webkit-flex; /* Safari */
    -webkit-flex-wrap: wrap; /* Safari 6.1+ */
    display: flex;
    flex-wrap: wrap;
         }

#main div {
    width: 50px;
    height: 50px;
}
</style>
</head>
<body>

<div id="main">
  <div>A</div>
  <div>B</div>
  <div>C</div>
  <div>D</div>
  <div>E</div>
  <div>F</div>
</div>
```



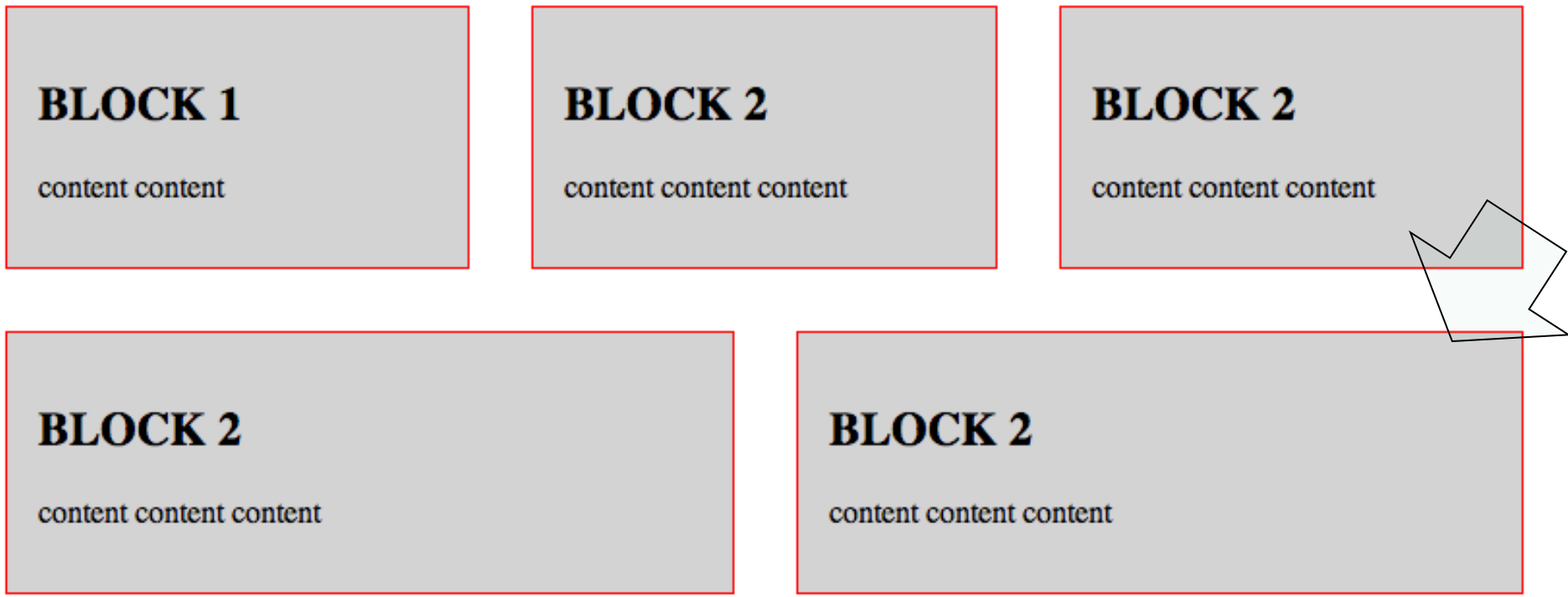
Note: Internet Explorer 10 and earlier versions do not support the flex-wrap property.

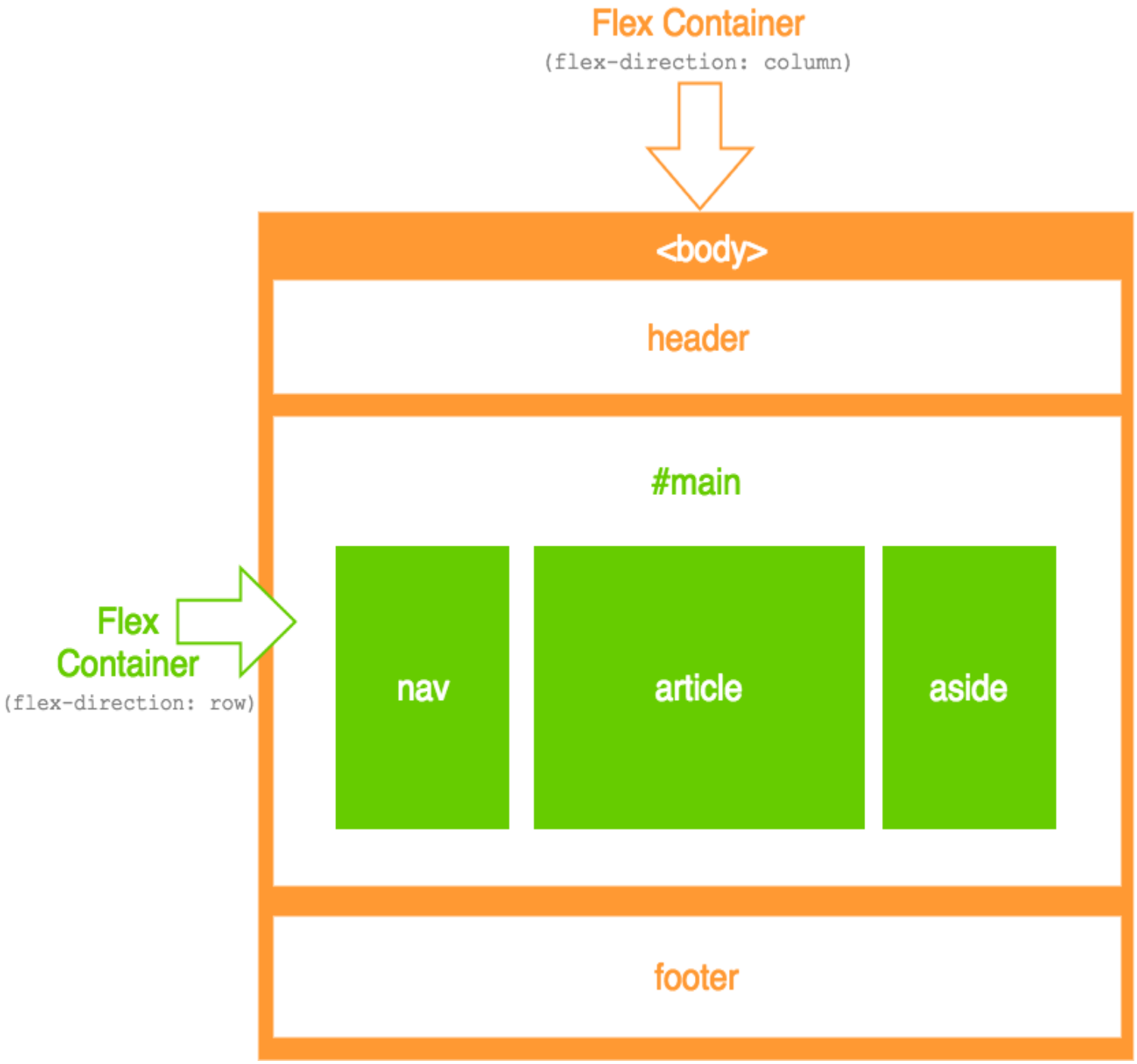
Note: Safari 6.1 (and newer) supports an alternative, the -webkit-flex-wrap property.

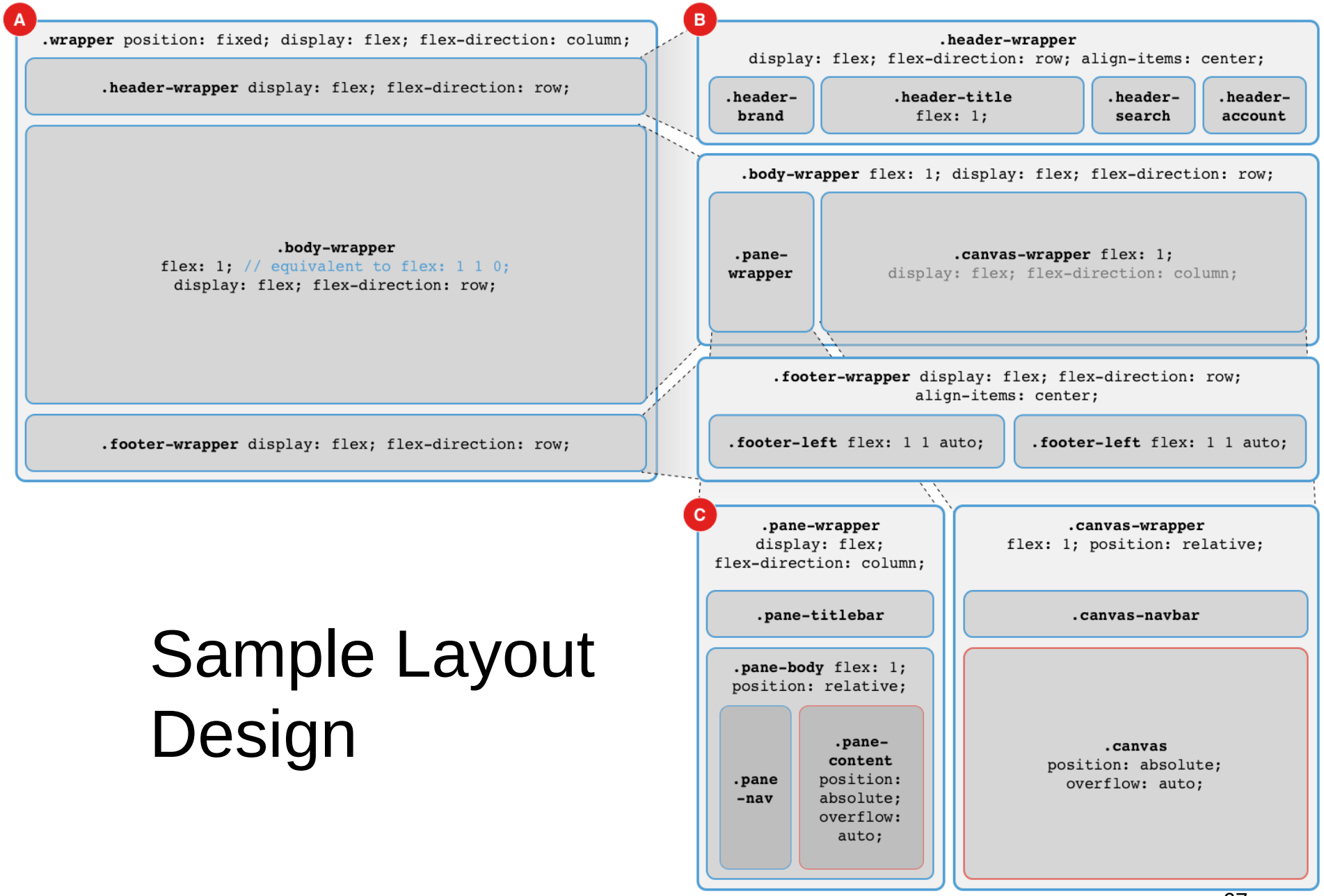
Usually desired: style flex items to stretch (and then wrap when a min-width occurs)

- Flex item (child) blocks to stretch: with third argument:
  - Minimum width to trigger wrapping …

```
#column_container section {
        -webkit-flex: 1 0 200px;
        flex: 1 0 200px;                    }
```




Flex Container
(flex-direction: column)
<body>
header
#main
Flex
Container
(flex-direction: row)
nav
article
aside
footer



Sample Layout Design

Breaking it down

```
.gallery-item {
  flex: 1 0 200px;
```

flex-grow

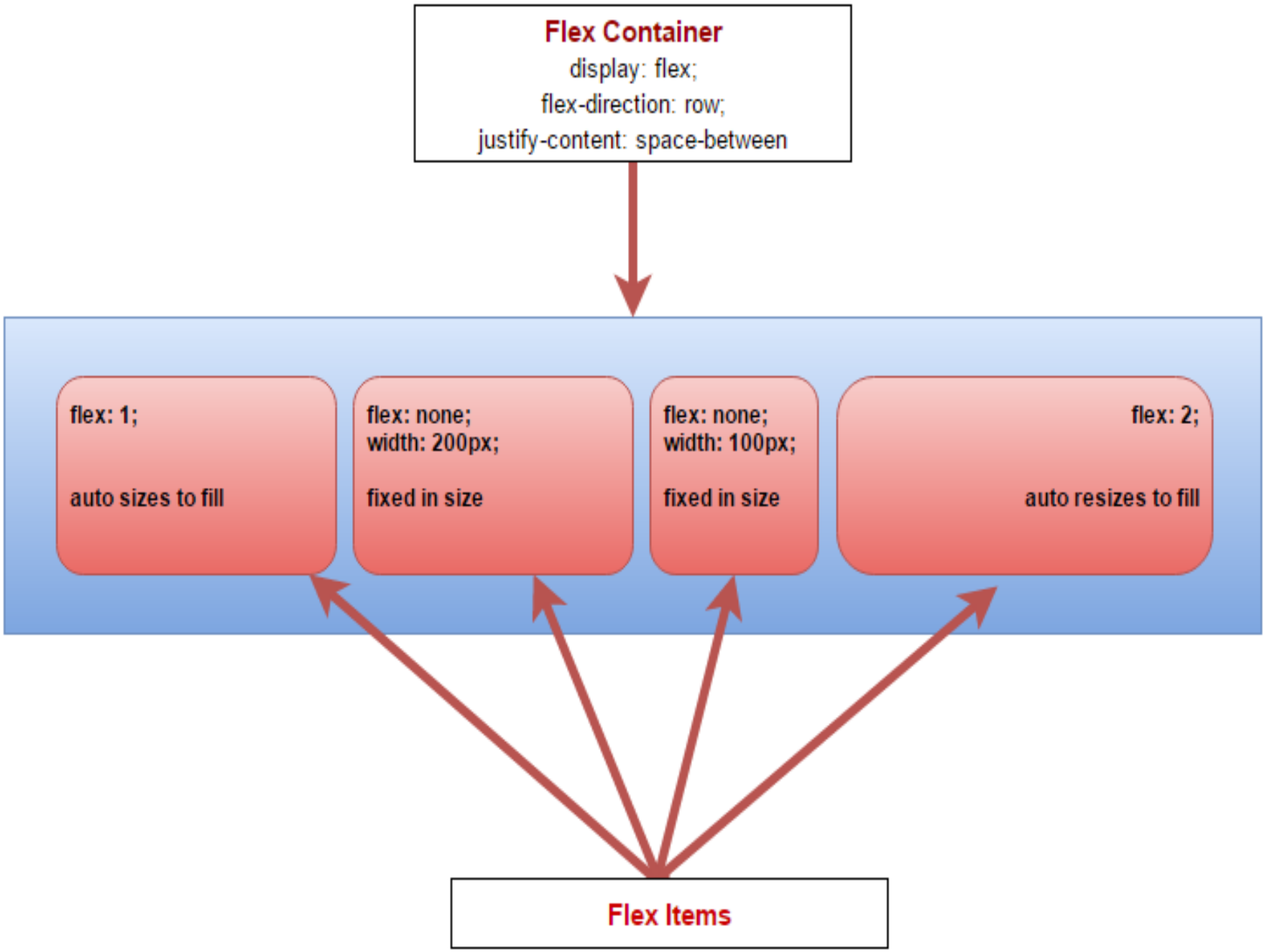
give every item 1 share of extra width

flex-shrink

don't let the items shrink at all (but they wouldn't anyway due to `flex-wrap`)

flex-basis

start them out at 200 pixels wide (basically, min-width)


Flex Container
display: flex;
flex-direction: row;
justify-content: space-between
flex: 1;
auto sizes to fill
flex: none;
width: 200px;
fixed in size
flex: none;
width: 100px;
fixed in size
flex: 2;
auto resizes to fill
Flex Items

Flex-grow

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

.flex-item.section{
  flex-grow: 1;
}

.flex-item.section{
  flex-grow: 2;
}
```

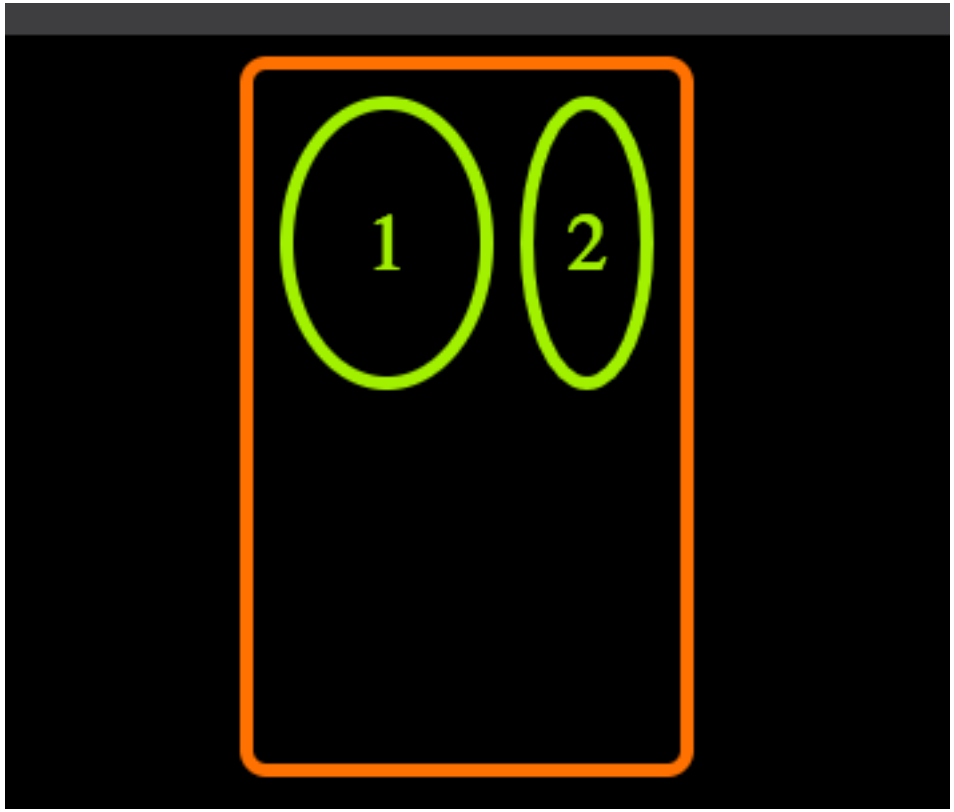
Flex-shrink

- flex-shrink only takes effect when the flex items overflow the flex container because of insufficient space.

- It is the ratio of overflowed width/height that the item will reduce to fit exactly the container's size.

- When being set to 0, the flex item will reduce 0% of the overflowed space, meaning it will not shrink at all.

- In this example, the ratio is 1:2,
- meaning the first item will reduce
- ⅓, while the second item will reduce
- ⅔ of the extra space.

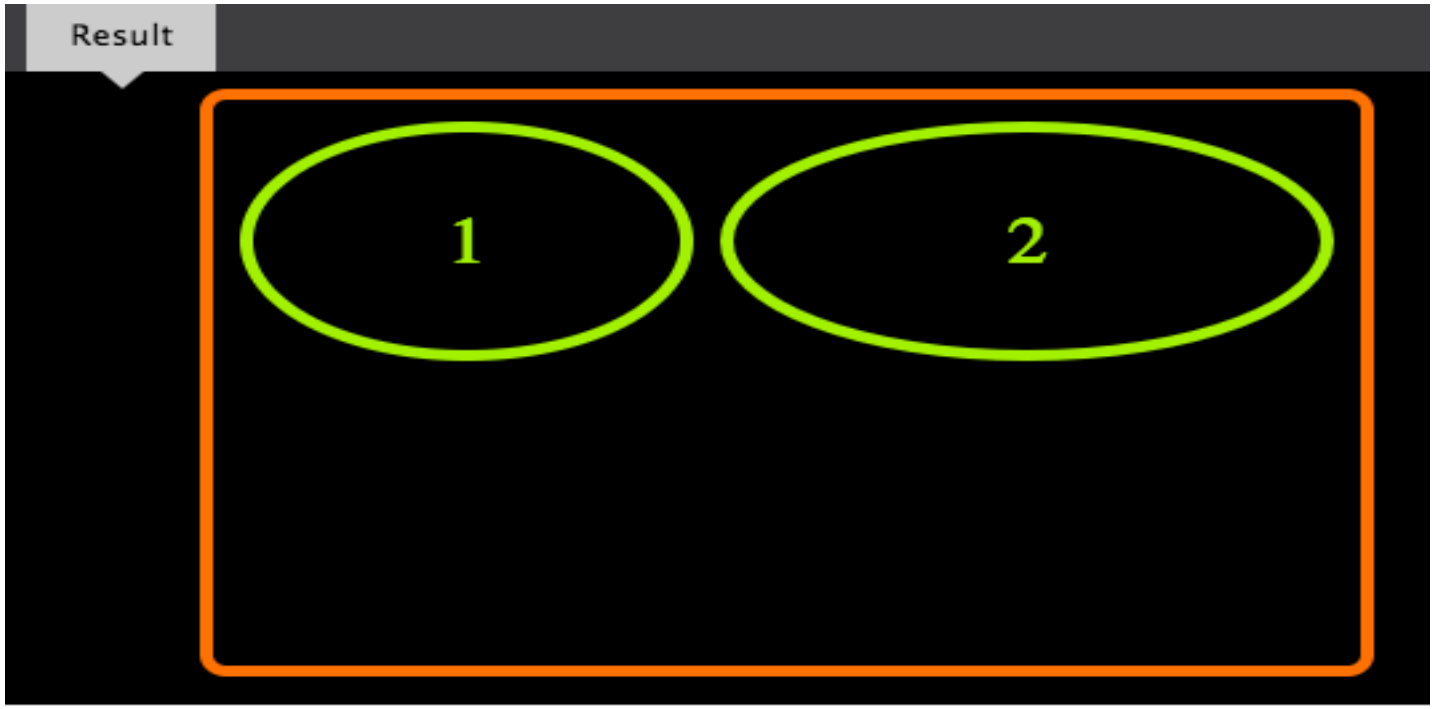


Expand elements

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

.flex-item.section{
  flex-grow: 1;
}

.flex-item.section {
  flex-grow: 2;
}
```

Flex-grow

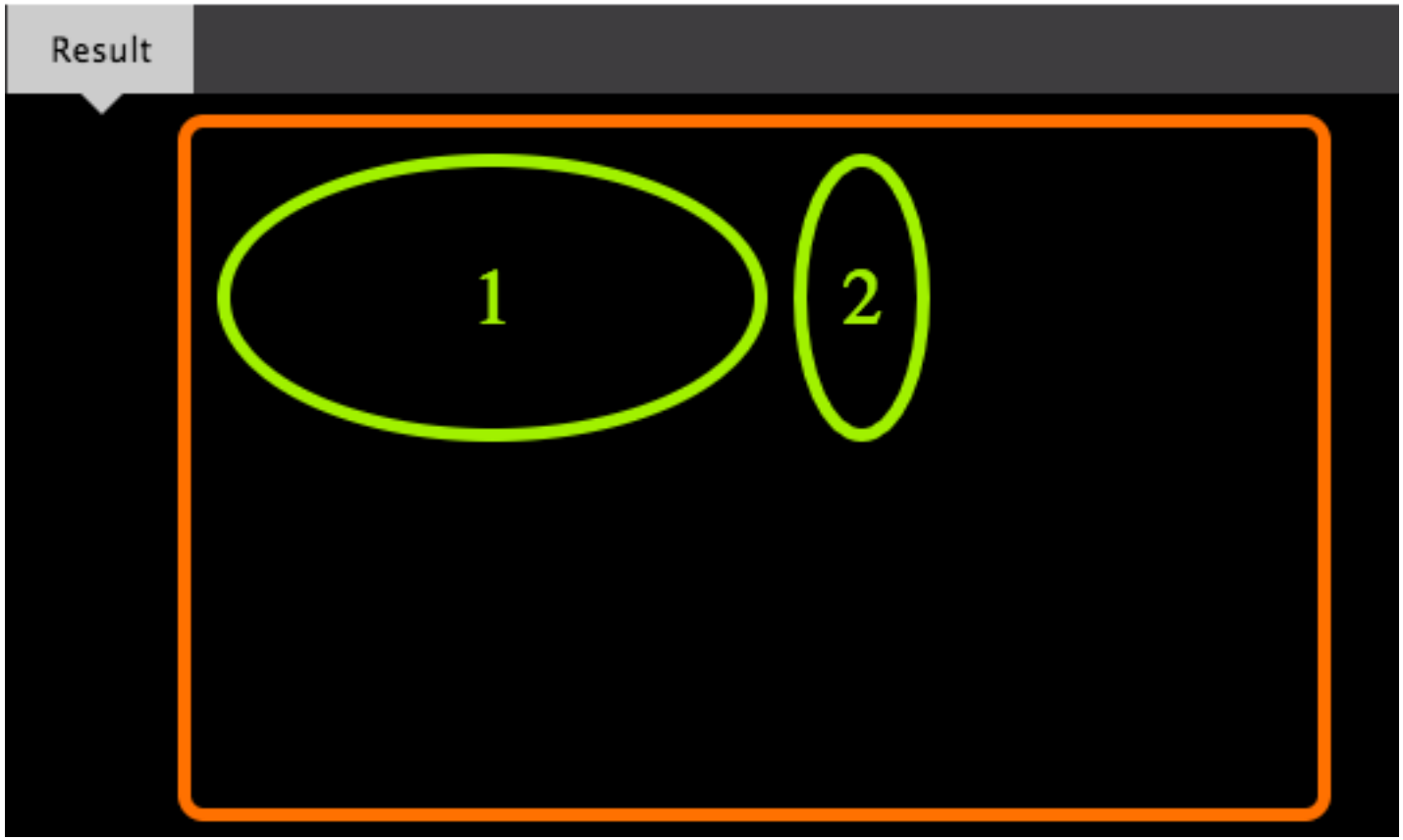
- flex-grow takes effect only when there is remaining space in a flex container.
- flex-grow of a flex item is the ratio of container's remaining width/height that the item takes to fit exactly the container's size.
- Default is set to be 1.
- When being set to 0, the flex item will take 0% of the remaining space, meaning it will not grow at all.
- In this example, the ratio is 1:2, meaning the first item will take up ⅓, while the second item will take ⅔ of the space left.



flex-basis

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

.flex-item.section {
  flex-basis: 200px;
}

.flex-item.section {
  flex-basis: 10%;
}
```

- Instead of using the initial size of an element, you can customize its size with flex-basis.
- By default the value is flex-basis: auto, in this case the initial size calculated from non-flexbox css rules are used.
- You can also set it to some absolute value and percentage value with respect to the flex container, for example flex-basis: 200px and flex-basis: 10%.



Put flex-grow, flex-shrink, flex-basis together

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

.flex-item:section {
  flex: 1 0 100px;
}

.flex-item:section {
  flex: 2 0 10%;
}
```

We will do this in next weeks lab

- Some handy uses of flexible boxes……..

Header/footer – left right alignment

- Flexible container
- 2 flex items
- Text align first one LEFT (default)
- Text align second one RIGHT

CSS flexible boxes

- BLOCK level elements
  - Basics are just 2 steps

(1) Flex 'container'

```
display: flex;
```

(2) Flex 'items' (children of container)

- In most cases we wish to make items equal width
- (and 'stretch' to fill available width of flex container)

```
flex: 1;
```

Other issues
wrap items

- 2 steps for wrapping blocks for narrow widths:

(1) for flex container add:

```
flex-wrap: wrap;
```

(2) for flex item need to specify min-width to trigger wrap

```
flex: 1 0 200px;
```

Other issues
ITB older Chrome versions – need –webkit-

```
display: flex;
display: -webkit-flex;
flex-wrap: wrap;
-webkit-flex-wrap: wrap;
flex-direction: column;
-webkit-flex-direction: column;
align-items: center;
-webkit-align-items: center;
```

Flex container properties

```
flex: 1;
-webkit-flex: 1;
flex: 1 0 10em;
-webkit-flex: 1 0 10em;
```

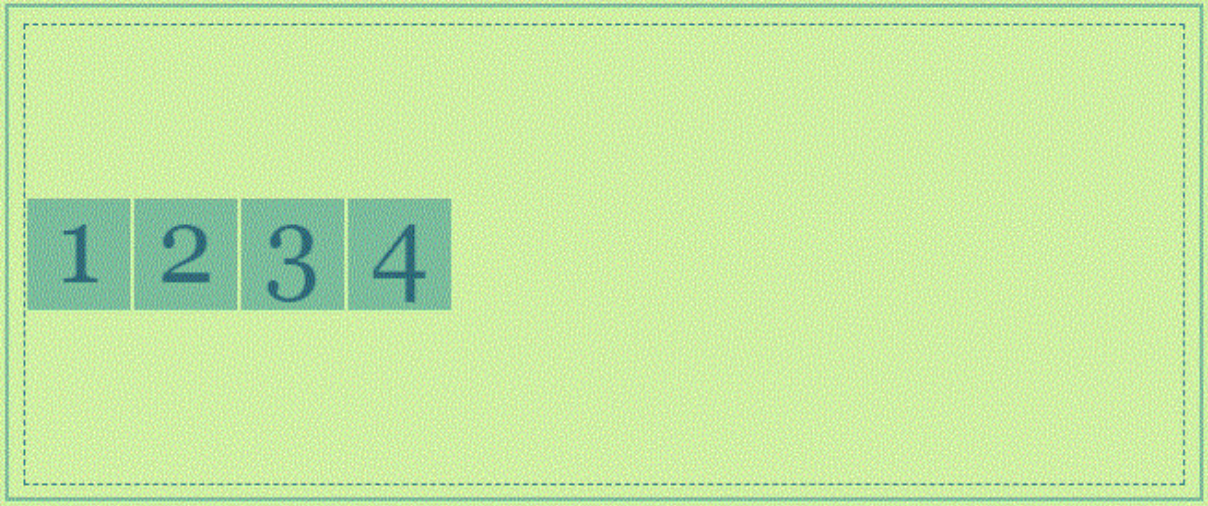
Flex item properties

Aligning items in the flex container

align-items

w3.org/TR/css-flexbox-1/#propdef-align-items

flex-start | flex-end | center | baseline | stretch




```
.parent {
  display: flex;
  align-items: center;
  height: 100%;
}
```

Flex items can be aligned in the *cross axis* of the current line of the flex container, similar to *justify-content* but in the perpendicular direction. *align-items* sets the default alignment for all of the flex container's *items*, including anonymous *flex items*. *align-self* allows this default alignmer to be overridden for individual *flex items*. (For anonymous flex items, *align-self* always matches the value of *align-items* on their associated flex container.)

align-items

w3.org/TR/css-flexbox-1/#propdef-align-items

flex-start | flex-end | center | baseline | stretch


```
.parent {
  display: flex;
  align-items: flex-end;
  height: 100%;
}
```

Flex items can be aligned in the *cross axis* of the current line of the flex container, similar to *justify-content* but in the perpendicular direction. *align-items* sets the default alignment for all of the flex container's *items*, including anonymous *flex items*. *align-self* allows this default alignment to be overridden for individual *flex items*. (For anonymous flex items, *align-self* always matches the value of *align-items* on their associated flex container.)

align-items

w3.org/TR/css-flexbox-1/#propdef-align-items

flex-start | flex-end | center | baseline | **stretch**


```
.parent {
  display: flex;
  align-items: stretch;
  height: 100%;
}
```

Flex items can be aligned in the *cross axis* of the current line of the flex container, similar to *justify-content* but in the perpendicular direction. *align-items* sets the default alignment for all of the flex container's *items*, including anonymous *flex items*. *align-self* allows this default alignment to be overridden for individual *flex items*. (For anonymous flex items, *align-self* always matches the value of *align-items* on their associated flex container.)

flex-direction

w3.org/TR/css-flexbox-1/#flex-direction-property

row | row-reverse | column | column-reverse


```
.parent {
  display: flex;
  flex-direction: row;
  height: 100%;
}
```

flex-grow

w3.org/TR/css-flexbox-1/#flex-grow-property

0 1


```
.parent {
  display: flex;
  height: 100%;
}
.child--featured {
  flex-grow: 1;
}
```

The *flex-grow* property sets the *flex grow factor* to the provided *<number>*. Negative numbers are invalid.

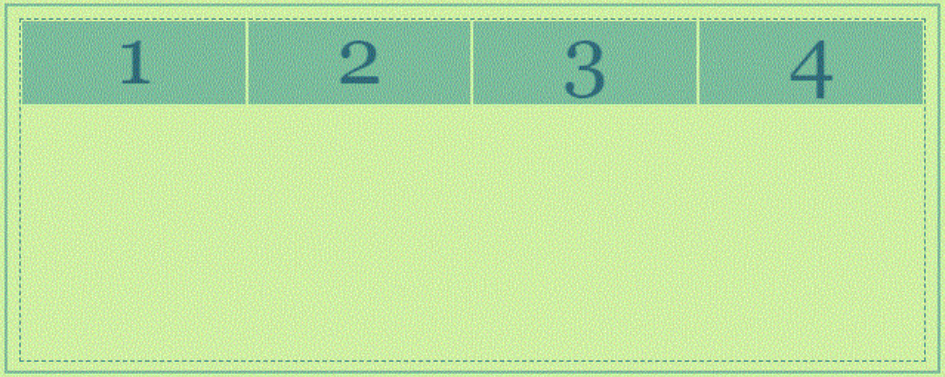
Applies to: flex items.

Initial: 0.

flex-wrap

w3.org/TR/css-flexbox-1/#flex-wrap-property

nowrap | wrap | wrap-reverse

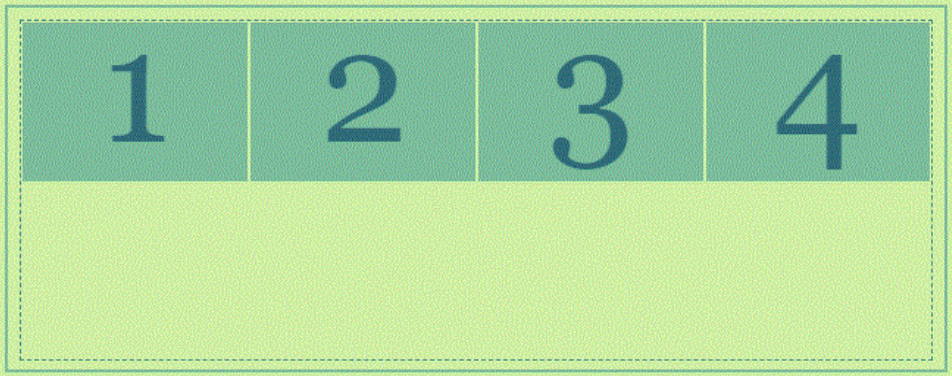



```
.parent {
  display: flex;
  align-items: flex-start;
  flex-wrap: nowrap;
  height: 100%;
}
.child {
  width: 40%;
}
```

flex-flow

w3.org/TR/css-flexbox-1/#flex-flow-property

row nowrap | column-reverse | column wrap | row-reverse wrap-reverse

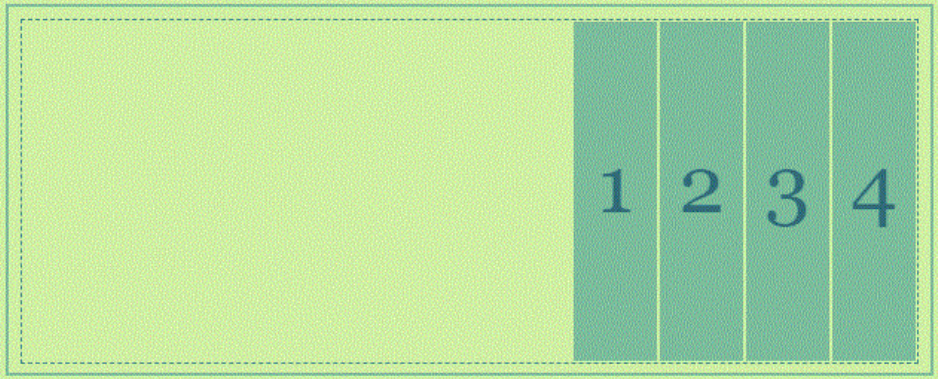



```
.parent {
  display: flex;
  flex-flow: row nowrap;
  height: 100%;
}
.child {
  width: 40%;
  height: 40%;
}
```

justify-content

w3.org/TR/css-flexbox-1/#justify-content-property

flex-start | **flex-end** | center | space-between | space-around | space-evenly



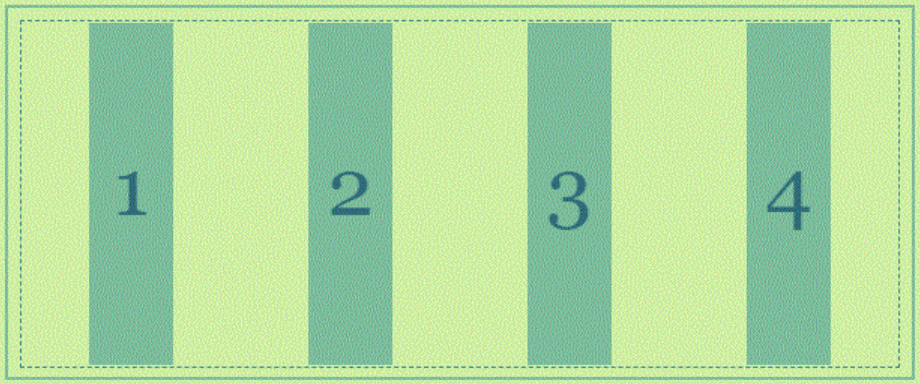

```
.parent {
  display: flex;
  justify-content: flex-end;
  height: 100%;
}
```

The *justify-content* property aligns *flex items* along the *main axis* of the current line of the flex container. This is done *after* any flexible lengths and any *auto margins* have been resolved. Typically it helps distribute extra free space leftover when either all the *flex items* on a line are inflexible, or are flexible but have reached their maximum size. It also exerts some control over the alignment of items when they overflow the line.

justify-content

w3.org/TR/css-flexbox-1/#justify-content-property

flex-start | flex-end | center | space-between | **space-around** | space-evenly



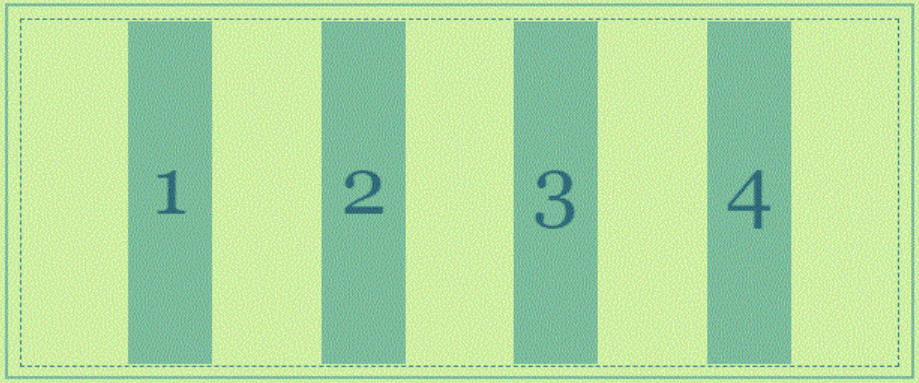

```
.parent {
  display: flex;
  justify-content: space-around;
  height: 100%;
}
```

The *justify-content* property aligns *flex items* along the *main axis* of the current line of the flex container. This is done *after* any flexible lengths and any *auto margins* have been resolved. Typically it helps distribute extra free space leftover when either all the *flex items* on a line are inflexible, or are flexible but have reached their maximum size. It also exerts some control over the alignment of items when they overflow the line.

justify-content

w3.org/TR/css-flexbox-1/#justify-content-property

flex-start | flex-end | center | space-between | space-around | space-evenly



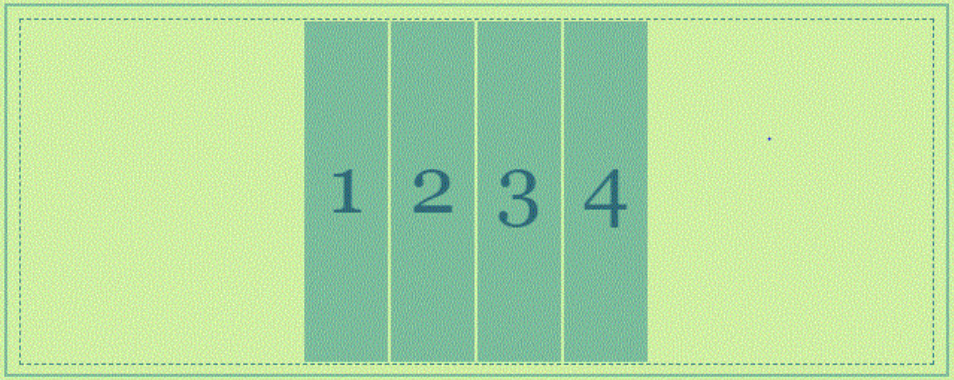

```
.parent {
  display: flex;
  justify-content: space-evenly;
  height: 100%;
}
```

The *justify-content* property aligns *flex items* along the *main axis* of the current line of the flex container. This is done *after* any flexible lengths and any *auto margins* have been resolved. Typically it helps distribute extra free space leftover when either all the *flex items* on a line are inflexible, or are flexible but have reached their maximum size. It also exerts some control over the alignment of items when they overflow the line.

justify-content

w3.org/TR/css-flexbox-1/#justify-content-property

flex-start | flex-end | center | space-between | space-around | space-evenly



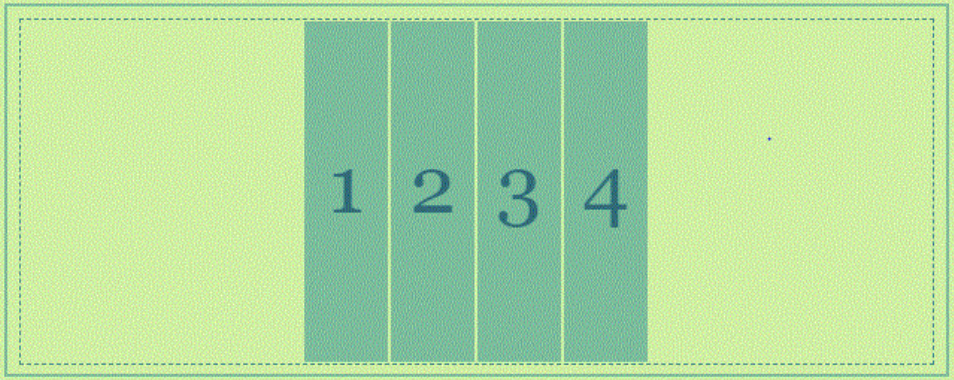

```
.parent {
  display: flex;
  justify-content: center;
  height: 100%;
}
```

The *justify-content* property aligns *flex items* along the *main axis* of the current line of the flex container. This is done *after* any flexible lengths and any *auto margins* have been resolved. Typically it helps distribute extra free space leftover when either all the *flex items* on a line are inflexible, or are flexible but have reached their maximum size. It also exerts some control over the alignment of items when they overflow the line.

justify-content

w3.org/TR/css-flexbox-1/#justify-content-property

flex-start | flex-end | center | space-between | space-around | space-evenly



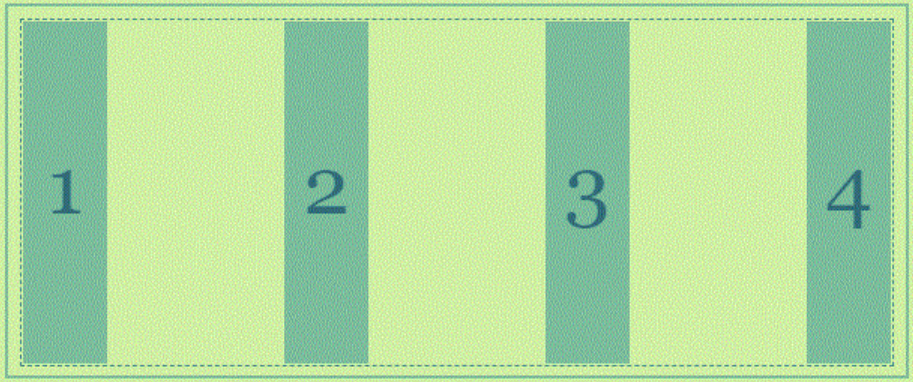

```
.parent {
  display: flex;
  justify-content: center;
  height: 100%;
}
```

The *justify-content* property aligns *flex items* along the *main axis* of the current line of the flex container. This is done *after* any flexible lengths and any *auto margins* have been resolved. Typically it helps distribute extra free space leftover when either all the *flex items* on a line are inflexible, or are flexible but have reached their maximum size. It also exerts some control over the alignment of items when they overflow the line.

justify-content

w3.org/TR/css-flexbox-1/#justify-content-property

flex-start | flex-end | center | **space-between** | space-around | space-evenly




```
.parent {
  display: flex;
  justify-content: space-between;
  height: 100%;
}
```

The *justify-content* property aligns *flex items* along the *main axis* of the current line of the flex container. This is done *after* any flexible lengths and any *auto margins* have been resolved. Typically it helps distribute extra free space leftover when either all the *flex items* on a line are inflexible, or are flexible but have reached their maximum size. It also exerts some control over the alignment of items when they overflow the line.

Remember

- Flex the header
- Flex the navigation bar
- Create a container – display flex
- Create the sections i.e <section><aside> <main> <article> for the main content of the page
- Add the above to the container
- Flex the footer
- Wrap and watch it resize

Header/footer – left right alignment

- Flexible container
- 2 flex items
- Text align first one LEFT (default)
- Text align second one RIGHT

wrapper

- Add the header, nav, container and footer to a wrapper
-
- See example over

South Africa South Africa

Home | Cuisine | Gallery | Contact

South Africa - The experience

It is a long established fact that a reader will be distracted by the readable content of a page when looking at its

It is a long established fact that a reader will be distracted by the readable content of a page when looking at its

Sample flex sites

The Envato Tuts+ Report

Read tutorial

Hello World

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Proin ornare magna eros, eu pellentesque tortor vestibulum ut. Maecenas non massa sem. Etiam finibus odio quis feugiat facilisis. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Proin ornare magna eros.

By Joe Smith 42 comments

Hello World

Lorem ipsum dolor sit amet, consectetur adipiscing elit.
Lorem ipsum dolor sit amet, consectetur adipiscing elit.
Lorem ipsum dolor sit amet, consectetur adipiscing elit.

By Joe Smith 42 comments

Hello World

Lorem ipsum dolor sit amet, consectetur

Hello World

Lorem ipsum dolor sit amet, consectetur

Hello World

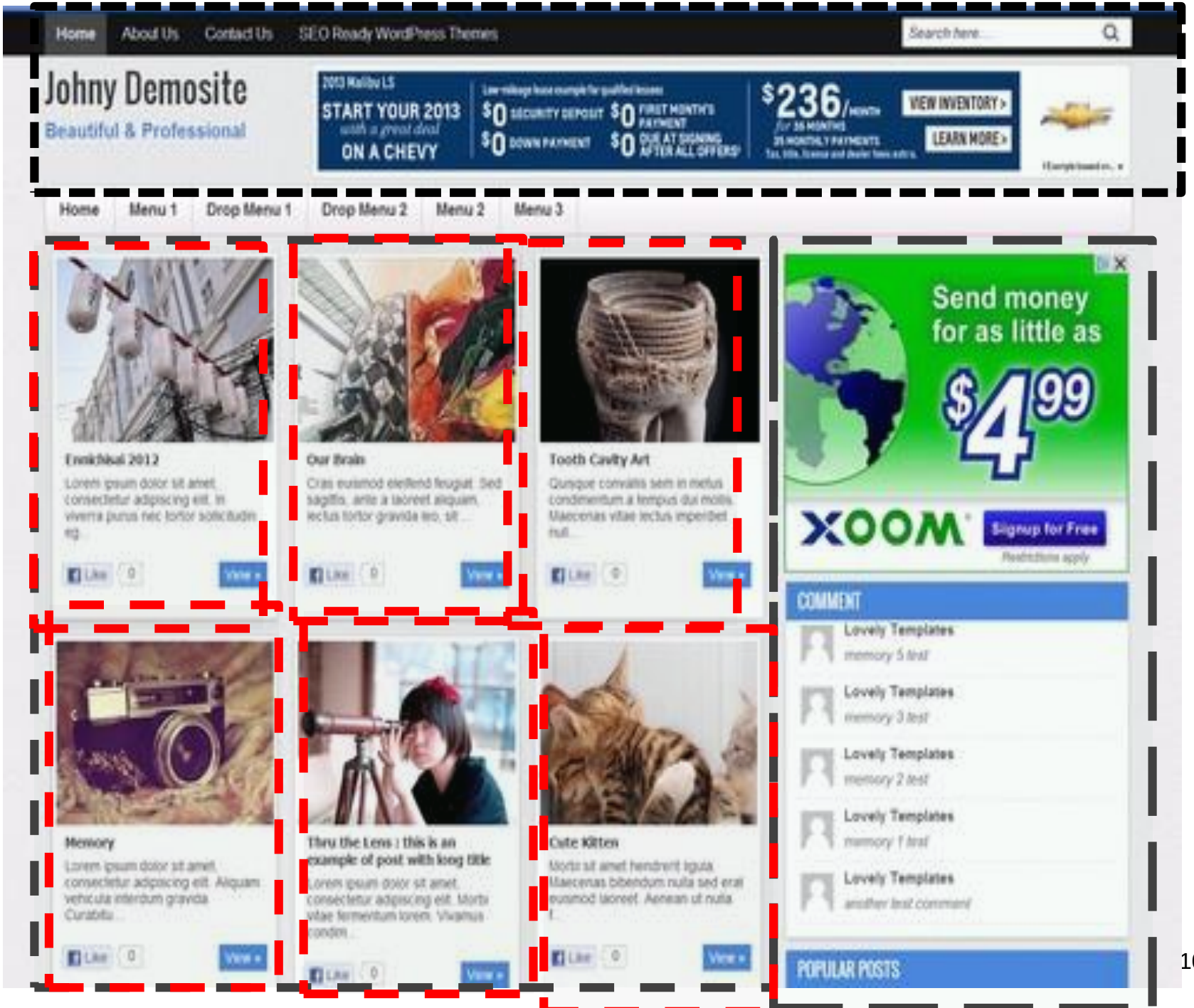
Lorem ipsum dolor sit amet, consectetur adipiscing elit.

By Joe Smith 42 comments

Flex

Flex

Flex 1 Each section gets even space


Home
About Us
Contact Us
SEO Ready WordPress Themes
Search here...
Johny Demosite
Beautiful & Professional
START YOUR 2013
with a great deal
ON A CHEVY
$0 SECURITY DEPOSIT
$0 FIRST MONTH'S PAYMENT
$0 DOWN PAYMENT
$0 DUE AT SIGNING AFTER ALL OFFERS!
$236/MONTH
VIEW INVENTORY
LEARN MORE
Home
Menu 1
Drop Menu 1
Drop Menu 2
Menu 2
Menu 3
Enrichkui 2012
Our Brain
Tooth Cavity Art
Send money for as little as
$4.99
XOOM
Signup for Free
COMMENT
Lovely Templates
Lovely Templates
Lovely Templates
Lovely Templates
Lovely Templates
Memory
Thru the Lens : this is an example of post with long title
Cute Kitten
POPULAR POSTS

Jan 26, 2018

Home About Contact

PAPERMAG

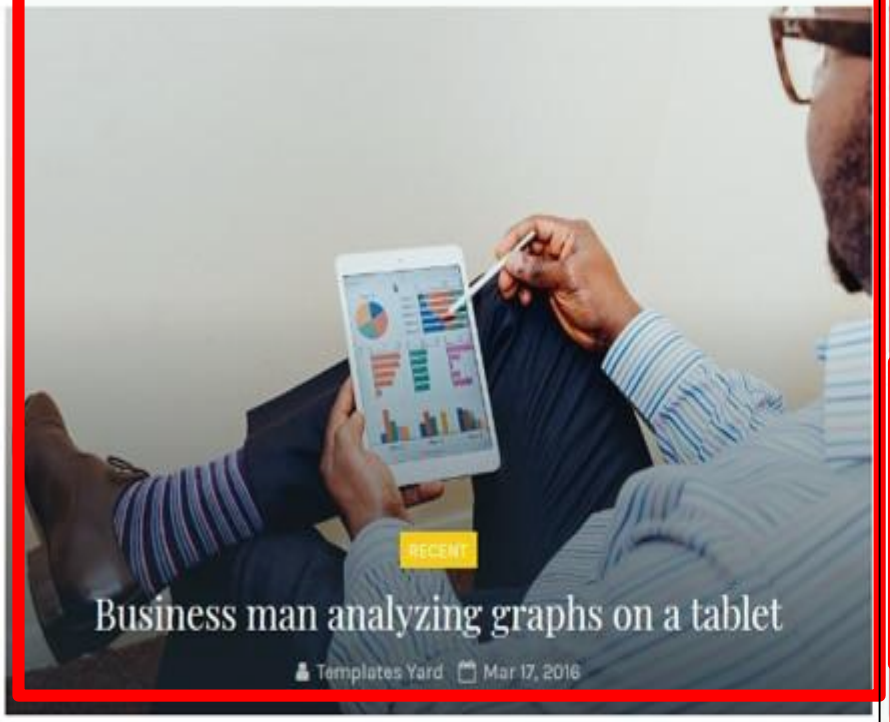
Stylish Magazine

HOME FEATURES SEO SERVICES DOCUMENTATION DOWNLOAD THIS TEMPLATE

RECENT

Business man analyzing graphs on a tablet

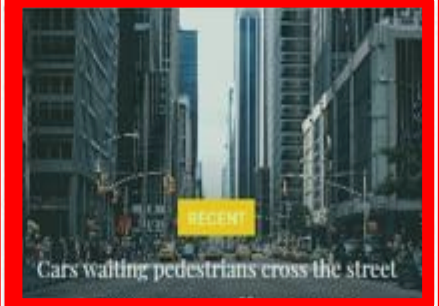
Mar 17, 2016



RECENT

Cars waiting pedestrians cross the street

Mar 17, 2016



RECENT

Tablet on a table showing calendar

Mar 17, 2016

RECENT

Business man reading newspaper

Mar 17, 2016

RESPONSIVE ADS HERE

FYI- Flexible boxes are W3C recommended layout format

- So (as with any part of computing) you need to keep up with web new standards as they are published

- Sources of flexible box information:

w3c candidate recommendation (2012)

dev.w3.org/csswg/css-flexbox/

w3c EDITOR'S draft (2013)

dev.w3.org/csswg/css-flexbox/

CSS tricks website (2013)

css-tricks.com/snippets/css/a-guide-to-flexbox/

Next weeks lab

1. Create a simple 2 page website with liquid/elastic layout
2. Create the same simple 2 page website using flex boxes
3. Complete a very simple survey to decide which method you prefer