Introducing CSS Flexbox Layout

PRESENTED BY

Matthew Ellison, UA Europe
MATTHEW ELLISON

• Consultant and trainer for User Assistance tools and technologies since 1993
• User of MadCap products since 2006
• Certified Flare Instructor
• Based in the UK
WHAT WE’LL COVER THIS AFTERNOON

- The limitations of html’s block model
- Responsive layout with Media Queries
- Introducing Flexbox
- Scenarios where Flexbox is useful
- Flexbox properties: building a solution from the ground up
- Browser support
- Implementing CSS Flexbox in Flare
THE LIMITATIONS OF HTML’S BLOCK MODEL

- h1
- div
- p
- p [img]
SIDE-BY-SIDE LAYOUT TRADITIONALLY SOLVED BY...

- Tables (deprecated for layout)
- Float property
ELEMENTS RENDERED IN NORMAL FLOW

h1

div

p

p
FLOAT PROPERTY

CSS:
float: left

css:
clear: both
EXAMPLE OF FLOAT-BASED LAYOUT
WHAT ABOUT CSS POSITIONING?

• Introduced in CSS2
• `position` property:
  - static
  - relative
  - absolute
  - fixed
• Not easy to understand or work with
RESPONSIVE LAYOUT WITH MEDIA QUERIES

Column 1 text column 1 text column 1 text
Column 1 text column 1 text column 1 text
Column 2 text column 2 text column 2 text
Column 2 text column 2 text column 2 text
VIEWPORT LESS WIDE

Heading text heading text heading text

Column 1 text column 1 text column 1 text column 1 text column 1 text

Column 2 text column 2 text column 2 text column 2 text column 2 text

42% float: left

16% 16% 16% 100%

16% 16% 16% 100%
### PAST BREAKPOINT

<table>
<thead>
<tr>
<th>Column 1 text column 1 text column 1 text column 1 text column 1 text column 1 text</th>
<th>Column 2 text column 2 text column 2 text column 2 text column 2 text column 2 text</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Heading text</strong></td>
<td><strong>100% float: none</strong></td>
</tr>
<tr>
<td><strong>16%</strong></td>
<td><strong>16%</strong></td>
</tr>
</tbody>
</table>
INTRODUCING CSS FLEXBOX

- Creates intelligent boxes that are stretchable, squeezable and capable of changing visual order
- Simple solution for layout paradigms that CSS has always struggled with:
  - vertical centering
  - equal heights
Within a flex container, items typically:
- Arrange themselves in rows or columns
- Expand to fill available free space
  or
- Shrink to prevent overflow

Flex container's margins do NOT collapse with the margins of its content.
A Visual Demonstration of Flexbox principles...
<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>D</td>
<td>E</td>
<td>F</td>
</tr>
<tr>
<td></td>
<td>G</td>
<td></td>
</tr>
</tbody>
</table>
NARROW SCREEN (PHONE)
FLEXBOX SCENARIOS

• Vertical centering
• Unknown number of items within a container (photos, landing page, toolbar/menu items)
• Flexible layout for website components (header, sidebar, main, footer)
VERTICAL CENTERING

Vertical centering without flexbox is hopeful at best

Demo by David Walsh at davidwalsh.name/css-vertical-center-flexbox
CONTINUOUS REARRANGEMENT OF ITEMS WITHIN A CONTAINER AS SCREEN SIZE CHANGES

UA Europe speakers at
www.uaconference.eu/speakers.html
FLEXIBLE LAYOUT FOR WEBSITE COMPONENTS

Demo by Karen Menezes at http://codepen.io/imohkay/details/JoYoRE
BASIC SPECIFICATIONS THAT WE CAN SPECIFY USING CSS

- Arrangement of items (row or column)
- ...and whether they are allowed to wrap
BASIC SPECIFICATIONS THAT WE CAN SPECIFY USING CSS

- Base width of items
  (used to calculate how many items can be fitted within each row)
BASIC SPECIFICATIONS THAT WE CAN SPECIFY USING CSS

- Whether the width of items is allowed to grow to fill the available space
BASIC SPECIFICATIONS THAT WE CAN SPECIFY USING CSS

- How items are justified within the container if they are not allowed to grow

```flex-start
A B C
D
```
BASIC SPECIFICATIONS THAT WE CAN SPECIFY USING CSS

- How items are justified within the container if they are not allowed to grow

```
ABC

D
```
BASIC SPECIFICATIONS THAT WE CAN SPECIFY USING CSS

- How items are justified within the container if they are not allowed to grow

```css
    space-between
```

```
A  B  C

D
```
Building the Flexbox CSS Code...
<div id="container">
  <div class="item">
    Content
  </div>
  <div class="item">
    Content
  </div>
  <div class="item">
    Content
  </div>
  <div class="item">
    Content
  </div>
</div>
CSS - CONTAINER PROPERTY: **FLEX**

```css
div#container {
    display: flex;
}

div.item {
}
```

The only property that you **need** to specify

Default behavior:
- Arrangement by row
- No wrap
- Justified to start of flex container
- Items shrink to fit within container
- Items do not grow to fill available space
CONTAINER PROPERTY: FLEX-DIRECTION

```
div#container
{
    display: flex;
    flex-direction: column;
}

div.item
{
}
```
CONTAINER PROPERTY: FLEX-WRAP

```css
div#container {
  display: flex;
  flex-direction: row;
  flex-wrap: wrap;
}

div.item {
}
```
CONTAINER PROPERTY: **justify-content**

```html
div#container
{
    display: flex;
    flex-direction: row;
    flex-wrap: wrap;
    justify-content: space-between;
}

div.item
{
}
```
ITEM PROPERTY: **FLEX-GROW**

```html
div#container
{
    display: flex;
    flex-direction: row;
    flex-wrap: wrap;
    justify-content: space-between;
}

div.item
{
    flex-grow: 1;
}
```

All items within a row grow by the same amount to fill the available space.

This property is now redundant.
ITEM PROPERTY: **FLEX-BASE**

```css
div#container
{
  display: flex;
  flex-direction: row;
  flex-wrap: wrap;
  justify-content: space-between;
}

div.item
{
  flex-grow: 1;
  flex-basis: 28%;
}
```

**container**

**item** | **item** | **item** | **item**

Used to calculate how many items will fit within a row

Items grow to fill row
FLEX-SHRINK PROPERTY

As long as flex-shrink is not 0, items will shrink if:

- an item is on its own in a row, and the browser width is less than the flex-basis property
  or
- if flex-wrap has a value of nowrap and the total of flex-basis properties for all items is greater than the width of the container
ALL CONTAINER FLEXBOX PROPERTIES

display: flex;

flex-direction: row | column | row-reverse | column-reverse;

flex-wrap: wrap | nowrap | wrap-reverse;

justify-content: flex-start | flex-end | center | space-between | space-around;

align-items: stretch | flex-start | flex-end | center | baseline;

align-content: stretch | flex-start | flex-end | center | space-between | space-around;
ALL ITEM FLEXBOX PROPERTIES

order: <number>; default is 0

flex-grow: <number>; default is 0 (it will not be allowed to grow)

flex-shrink: <number>; default is 1 (it will be allowed to shrink)

flex-basis: auto | <width/height in px or %>

(auto means it will be sized automatically based on its width property)

align-self: auto | flex-start | flex-end | center | baseline | stretch;
Landing Page

Select one of the following options:

- **Set up new users**
  This section describes the steps a Flare/TFS administrator needs to do when a new author joins the Flare authoring team.

- **Leaver tasks**
  This section describes the tasks that a Flare/TFS administrator needs to do when a user leaves.

- **Licensing**
  This section provides information on licensing the suite of MadCap Flare authoring tools.

- **TFS administration overview**
  This section provides information on managing team projects and user access and privilege in TFS.

- **Manage Flare project access**
  This section provides information on how you assign and manage access to Flare projects in TFS.

- **Manage global project**
  This section provides information on how to manage the global project including how to make it available to all users.

- **Upgrading**
  This section provides links to content that is relevant for when Flare or TFS is upgraded, or when any other system changes are implemented that impact the authoring environment.
ABSOLUTE FLEX

• flex-basis of 0:
  – a way of easily controlling relative widths of items
  – **ALL** space within the container is assigned to items according to their relative flex-grow values
  – width property of the items is ignored
ABSOLUTE FLEX

class container
{
  display: flex;
}

class item
{
  flex-basis: 0;
}

class item:nth-of-type(odd)
{
  flex-grow: 1;
}

class item:nth-of-type(2)
{
  flex-grow: 2;
}

class item:nth-of-type(4)
{
  flex-grow: 5;
}
W3C encourages authors to use the flex shorthand property for items
d shorthand “correctly resets any unspecified components to accommodate
common uses”

```
flex: <flex-grow> <flex-shrink> <flex-basis>
```

• For example:

```
flex: 1 1 auto
```
## COMMON VALUES

<table>
<thead>
<tr>
<th>Shorthand</th>
<th>Equivalent to</th>
</tr>
</thead>
<tbody>
<tr>
<td>flex: initial</td>
<td>flex: 0 1 auto</td>
</tr>
<tr>
<td>flex: auto</td>
<td>flex: 1 1 auto</td>
</tr>
<tr>
<td>flex: none</td>
<td>flex: 0 0 auto</td>
</tr>
<tr>
<td>flex: &lt;+ve number&gt;</td>
<td>flex: &lt;+ve number&gt; 1 0</td>
</tr>
</tbody>
</table>
**BROWSER SUPPORT**

- Flexbox is supported in all of the latest browsers

<table>
<thead>
<tr>
<th>Browser</th>
<th>Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chrome</td>
<td>29.0</td>
</tr>
<tr>
<td></td>
<td>21.0 -webkit-</td>
</tr>
<tr>
<td>Edge</td>
<td>12.0</td>
</tr>
<tr>
<td>Firefox</td>
<td>11.0</td>
</tr>
<tr>
<td></td>
<td>10.0 -ms-</td>
</tr>
<tr>
<td>Firefox</td>
<td>28.0</td>
</tr>
<tr>
<td></td>
<td>18.0 -moz-</td>
</tr>
<tr>
<td>IE</td>
<td>9.0</td>
</tr>
<tr>
<td></td>
<td>6.1 -webkit-</td>
</tr>
<tr>
<td>Opera</td>
<td>17.0</td>
</tr>
</tbody>
</table>

- If you need to support older browsers, check out [caniuse](https://caniuse.com/) to see what your options are
ISSUES WORKING WITH FLEXBOX IN FLARE

- Flex property not recognized

- Layout is not shown correctly in XML Editor (but is correct in Preview)
REAL LIFE IMPLEMENTATION OF CSS FLEXBOX IN FLARE

Features
This is a general introduction to all of the features.

Feature 1
This description of Feature 1 is single-sourced from the Description field in Topic Properties.

Feature 2
This description of Feature 2 is single-sourced from the Description field in Topic Properties.

Feature 3
This description of Feature 3 is single-sourced from the Description field in Topic Properties. Here is more text than the other descriptions and therefore increases the height of the containing box.

Feature 4
This description of Feature 4 is single-sourced from the Description field in Topic Properties.

Feature 5
This description of Feature 5 is single-sourced from the Description field in Topic Properties.

Feature 6
This description of Feature 6 is single-sourced from the Description field in Topic Properties.

Feature 7
This description of Feature 7 is single-sourced from the Description field in Topic Properties. It has more text than some of the other descriptions.

Items based on Menu proxy (generated automatically on build)
REFERENCES AND FURTHER READING

- https://www.w3.org/TR/css-flexbox/
- https://cvan.io/flexboxin5/ (Great!)
- https://css-tricks.com/snippets/css/a-guide-to-flexbox/
- http://caniuse.com/#feat=flexbox
- https://www.smashingmagazine.com/2015/03/harnessing-flexbox-for-todays-web-apps/
- https://scotch.io/tutorials/a-visual-guide-to-css3-flexbox-properties
Thanks for attending!
Final questions?
Matthew Ellison
matthew@uaeurope.com