PUTTING FLEXBOX INTO PRACTICE
I’m a web designer/front-end dev

and I wrote these books on CSS:

**Stunning CSS3:**
A Project-based Guide to the Latest in CSS

[www.stunningcss3.com](http://www.stunningcss3.com)

**Flexible Web Design:**
Creating Liquid and Elastic Layouts with CSS

[www.flexiblewebbook.com](http://www.flexiblewebbook.com)
Welcome to my little niche of the world wide web. Why have I chosen to add to the clutter of the information superhighway even more? It’s more than just self-indulgence, though I am hopelessly addicted to html and Photoshop. But besides having fun with web design, it is also the field I have worked in for some time now.

To view my online work, as well as examples of my visual artwork, click on gallery. For a more detailed description of my skills, experience, and goals, click on résumé. To learn more about me personally, click on me to the right. And finally, what self-respecting home page would be complete without a list of links for you to peruse?
**My portfolio site from 2000**

Welcome to my little niche of the world wide web. Why have I chosen to add to the clutter of the information superhighway even more? It's more than just self-indulgence, though I am hopelessly addicted to html and Photoshop. But besides having fun with web design, it is also the field I have worked in for some time now.

To view my online work, as well as examples of my visual artwork, click on gallery. For a more detailed description of my skills, experience, and goals, click on résumé. To learn more about me personally, click on me to the right. And finally, what self-respecting home page would be complete without a list of links for you to peruse?
2003: Introduced to floats
Problems with float layout

- Difficulty with containment
- Wrapping/float drop
- Visual location somewhat tied to HTML order
- No built-in equal-height columns
- No float:center
What is flexbox?

The nickname for the **CSS Flexible Box Layout Module**, a new layout mechanism and box model.
Which is which?

2009  display:box

2011  display:flexbox

Now  display:flex

Turn it on

display: flex

flex container

flex item

plain old box

flex item
**flex-direction**
specifies orientation of flex items’ layout

- **row** (default)
- **row-reverse**
- **column**
- **column-reverse**
Demo site:

Visit www.smoresday.us using Chrome, Opera, or IE 10 for full effect
Demo: horizontal navigation

1. Turn `<ul>` into flex container:

   ```css
   .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
What's the big deal?

I can do the same thing with display: inline."

Yes, you can.

We're just laying the groundwork for the cool stuff that's coming.

Baby steps.
How the CSS might really look

.list-nav {
  display: -webkit-box;
  display: -ms-flexbox;
  display: -webkit-flex;
  display: flex;
  -webkit-box-direction: normal;
  -webkit-box-orient: horizontal;
  -ms-flex-direction: row;
  -webkit-flex-direction: row;
  flex-direction: row;
}
Keeping track of variants

- Flexy Boxes code generator shows old and new syntaxes: [www.the-echoplex.net/flexyboxes/](http://www.the-echoplex.net/flexyboxes/)

- Let Sass or LESS do it for you, for instance:
  - [https://github.com/mastastealth/sass-flex-mixin](https://github.com/mastastealth/sass-flex-mixin)
  - [https://gist.github.com/cimmanon/4461470](https://gist.github.com/cimmanon/4461470)

- The -prefix-free script can do some of it for you: [http://leaverou.github.io/prefixfree/](http://leaverou.github.io/prefixfree/)
Use the variants you want.

But for the sake of readability, I’m omitting them from the code samples on these slides. You can see them in the live demo.

Now back to flex-direction.
Setting a point of reference

Main axis

Cross axis

(for flex-direction: row)
flex-wrap
controls whether flex items can lay out on multiple lines and which direction new lines are stacked in

- **wrap** (for row)
- **nowrap** (default; for row)
- **wrap-reverse** (for row)

wrap (for column)
Problems with flex-wrap

- Firefox doesn’t support it yet
- No browser supports 2009 equivalent box-lines property
- Limited control of where breaks occur without support for break-before/break-after properties (only IE 10 and Opera support them)
Summary: setting the stage

1. Create flex container using `display: flex`
2. Set its `flex-direction` to control orientation (horizontal or vertical)
3. Set its `flex-wrap` to control whether and in which direction to wrap
   (Or, set `flex-flow` as shorthand for `flex-direction` and `flex-wrap`)
Mobile block layout
Add flexbox for larger widths

No need to put within media query—it can “kick in” whenever space allows (auto breakpoints!)

1. Create flex container:
   
   ```
   .gallery {
       display: flex;
       flex-wrap: wrap;
       margin-left: -20px;
   }
   ```
Add flexbox for larger widths

2. Size flex items:

```css
.gallery-item {
  flex: 1 0 200px;
  margin: 0 0 20px 20px;
  padding: 20px;
}
```
This is where flexbox gets flexible.
And kinda confusing.
Defining the flex property

- Makes flex items change their width or height (whichever is dimension along main axis) to fit available space
- Shorthand for 3 properties:
  - `flex-grow`
  - `flex-shrink`
  - `flex-basis`
## Defining the flex property

<table>
<thead>
<tr>
<th>flex-grow</th>
<th>flex-shrink</th>
<th>flex-basis</th>
</tr>
</thead>
<tbody>
<tr>
<td>how much flex item will grow relative to other items if extra space is available (proportion of extra space that it gets)</td>
<td>how much item will shrink relative to others if there is not enough space (proportion of overflow that gets shaved off)</td>
<td>the initial starting size before free space is distributed (any standard width/height value, including auto)</td>
</tr>
</tbody>
</table>
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)
Let’s see it flex live
Single-digit flex values

- Common to see `flex: 1` in demos
- `flex: [number]` is equivalent to `flex: [number] 1 0px`
- Be sure you really want `flex-basis` to be `0`
  - When wrap on, essentially min-width
  - `0px` therefore means items can shrink to `0px`
  - If everything can get down to `0px`, nothing ever has a reason to wrap
My first attempt

Zoe’s Brain Said:
“Since .action starts out at 100%, it won’t have space to sit on the first line with the content preceding it, and will wrap to a second line.”

```css
.component {
  flex: 1;
}

.action {
  flex: 1 1 100%;
}
```

The expected outcome:
Flexbox fail

S'MORES BUILDER

This is your chance to get creative. As long as you have a roasted marshmallow sandwiched between something with some chocolate added, I say you got a s'mores. So pick your frame, marshmallow, and candy, add an optional accessory or two, and build a crazy s'mores concoction.

BUILD IT
My first attempt

Reality:
Since it’s fine for each .component to shrink to only 0px wide, a 100% wide element can and will sit on the same line as all the components.

.component {
  flex: 1 1 0px;
}

.action {
  flex: 1 1 100%;
}
Forcing the wrap

Fixed:
.action will always wrap to new line, and .components will wrap to additional lines when there’s less than their combined flex-basis values (plus margin, etc.).

```css
.component {
  flex: 1 1 200px;
}

.action {
  flex: 1 1 100%;
}
```
Live demo time again
Why flex is great
Less need for media queries

Layout changes that would previously have been hardcoded into a media query can now be done on the fly when browser determines stuff can fit
Flex adjusts for margin

box-sizing only takes care of padding and border added on to width, not margin

The boxes won’t all fit

```css
.component {
    width: 25%;
    margin-left: 20px;
}
```

Works like a charm

```css
.component {
    flex: 1 1 200px;
    margin-left: 20px;
}
```
Flex adjusts for quantity of items

- Great for sites with dynamic or frequently changing content blocks, e.g.:
  - News stories on home page vs inner page
  - Product or feature tiles
  - Pricing page with plans side by side
- Makes HTML/CSS more modular—an item can move anywhere and adjust in size as needed
Flex can combine different units

Items measured in different units can sit side-by-side and all fit perfectly

- Pixels
- Ems
- Mystery percentage
Flex can combine different units

Set only the text field to flex:

```
.component li:last-child {
  display: flex;
}
.component .other-name {
  flex: 1;
}
```
Flex can be proportional

Setting `flex-grow/flex-shrink` to different values can make flex items size themselves relative to each other.

```css
flex: 1;  flex: 1;  flex: 2;
```

![Diagram showing the effect of different flex values on flex items.]
But be careful!

Having widths be in multiples of each other only works if `flex-basis` is 0

```
flex: 1 0 0px;  flex: 1 0 0px;  flex: 2 0 0px;
```

If all start out 0px, then **all** the width on the line is extra, so the `flex: 2` item gets twice as much width as the others and is thus twice as wide as the others.
If flex-basis isn’t 0px…

…the widths may not end up as you expect

- `flex: 1 0 10px;`  
  - `10px + 5px extra = 15px`

- `flex: 1 0 10px;`  
  - `10px + 5px extra = 15px`

- `flex: 2 0 10px;`  
  - `10px + 10px extra = 20px`

if 50px available

The third box gets twice as much of the extra, but that doesn’t make it twice as wide overall
While support improves, consider using flexbox now on small page components as progressive enhancement.

Here are a few ideas.
Single-line, full-width form

- All items on same line
- Text input(s) stretches to fill remaining space
- All items vertically centered or equal height
Form without flexbox

```
.action { text-align: center; }
.action * {
    display: inline; /* default */
    vertical-align: middle;
}
```
Form without flexbox

- All items on same line
- Text input stretches to take up remaining space
- All items vertically centered or equal height

NAME YOUR S'MORES: [Input Field] [BUILD IT]

NEED SOME INSPIRATION?

CLASSIC
plain graham crackers
plain marshmallow
Hershey’s chocolate

ELVIS
chocolate graham crackers
plain marshmallow
Hershey’s chocolate
peanut butter
banana

PB&J
plain graham crackers
plain marshmallow
Hershey’s chocolate
peanut butter
strawberries

CHOCOLATE-COVERED
STRAWBERRY
chocolate graham crackers
plain marshmallow
dark chocolate
strawberries
Form with flexbox

```css
.action {
  flex: 1 1 100%;
  display: flex;
  align-items: stretch; /* default */
}

.action input {
  flex: 1;
}

.action input, .action label {
  margin-right: 10px;
}
```
align-items
aligns flex items in cross axis

- flex-start
- center
- flex-end
- baseline
Form with flexbox

✔ All items on same line
✔ Text input stretches to take up remaining space
✔ All items vertically centered or equal height
Combine the two

```
.action {
  flex: 1 1 100%;
  display: flex;
  text-align: center; /* fallback */
}

.action input {
  flex: 1;
}

.action label {
  align-self: center;
}

.action input, .action label {
  margin-right: 10px;
}
```
Another option: stack, center

```
.action {
  flex: 1 1 100%;
  display: flex;
  flex-wrap: wrap;
  align-items: center;
  text-align: center; /* fallback */
}
.action input {
  flex: 1;
  display: block; /* fallback */
  width: 100%; /* fallback */
  box-sizing: border-box; /* fallback */
}
.action button {
  flex: 1 1 100%;
  margin-top: 10px;
}
```
Narrow version

Non-flexbox fallback version

NAME YOUR S'MORES:

BUILD IT

NEED SOME INSPIRATION?

CLASSIC
plain graham crackers
plain marshmallow
Hershey’s chocolate

ELVIS
chocolate graham crackers
plain marshmallow
Hershey’s chocolate
peanut butter

Flexbox version

NAME YOUR S'MORES:

BUILD IT

NEED SOME INSPIRATION?

CLASSIC
plain graham crackers
plain marshmallow
Hershey’s chocolate

ELVIS
chocolate graham crackers
plain marshmallow
Hershey’s chocolate
peanut butter
Add Modernizr as needed

- Flexbox and fallback styles can often co-exist, but sometimes need to isolate them.
- Modernizr can add flexbox, no-flexbox, and flexbox-legacy classes to do this.
- Example: add margin between label and input only if flexbox is on:

```css
.flexbox .action label {
  margin-right: 10px;
}
```
Full-width nav bar

- Requirements:
  - All links on same line
  - First link flush left, last link flush right
  - Equal spaces between all links

- Using `display: table-cell` can do full-width but not equal spaces
Nav with flexbox

.list-nav {
    display: flex;
    justify-content: space-between;
    margin: 0;
    padding: 0;
    list-style: none;
}

.list-nav li {
    /* no flex & no width = shrinkwrap */
    text-align: center;
}
justify-content
aligns flex items along main axis

- flex-start (default)
- center
- flex-end
- space-between
- space-around
Combine with inline-block

.list-nav {
    display: flex;
    justify-content: space-between;
    margin: 0;
    padding: 0;
    list-style: none;
    text-align: center; /* fallback */
}

.list-nav li {
    display: inline-block; /* fallback */
    padding: 0 .5em; /* fallback */
    text-align: center;
}

.list-nav li:first-child { padding-left: 0; }
.list-nav li:last-child { padding-right: 0; }
Combine with inline-block

Non-flexbox fallback version

Flexbox version
Combine with table-cell

```
.list-nav {
    width: 100%; /* fallback */
    display: table; /* fallback */
    display: flex; /* override display:table for flexbox browsers */
    justify-content: space-between;
}

.list-nav li {
    display: table-cell; /* fallback */
    padding: 0 .5em;
    text-align: center;
}

.list-nav li:first-child {
    padding-left: 0;
    text-align: left; /* fallback */
}

.list-nav li:last-child {
    padding-right: 0;
    text-align: right;
}
```
Variation: pagination

- Wide view: all links on same line, centered
  - Set `justify-content: center`
- Medium view: all links on same line, full-width, equal spacing
  - Set `justify-content: space-between`
Variation: pagination

- Narrow view: two lines with "previous" and "next" links on first row, full-width
  - Set `flex-wrap: wrap`
  - Set `justify-content: space-between`
  - Use `order` property to move "next" link up
Visual reordering with flexbox

1. Make “previous” link come first visually, “next” link second, and all the rest third

```
.pagination li {
    order: 2;
    display: inline-block; /* fallback */
}

.pagination li:first-child { /* “Previous” link */
    order: 0;
    text-align: left;
}

.pagination li:last-child { /* “Next” link */
    order: 1;
    text-align: right;
}
```
Visual reordering with flexbox

2. Force links to wrap after “next” link by making it and “previous” link take up 100% of the first line together

```css
.flexbox .pagination li:first-child, .flexbox .pagination li:last-child {
  width: 50%;
}
```

```css
.pagination {
  display: flex;
  flex-wrap: wrap;
  justify-content: space-between;
  text-align: center; /* fallback */
}
```
## Accessibility implications

<table>
<thead>
<tr>
<th>Pro</th>
<th>Con</th>
</tr>
</thead>
<tbody>
<tr>
<td>Keep content in logical order in HTML instead of structuring HTML to achieve visual layout</td>
<td>Focus/tab order won’t always match expected order, may jump around seemingly randomly</td>
</tr>
</tbody>
</table>
Tab order = HTML order

“Next” won’t be second link tabbed to after “Previous” since it doesn’t follow it in HTML
Limitations of order property

- Potential to create confusing tab order
- Can only rearrange sibling elements
- Flexbox rows/cols can’t overlap, so content may not always slide into the spot left by the re-ordered content

So: reserve flexbox order property for small moves that don’t hurt accessibility, and use CSS3 Grid Layout, etc., for bigger re-ordering
Pinned item at bottom

- All boxes equal in height
- Final item in each box pinned to the bottom so that all final items across grid appear to align
Without flexbox, “other” fields disconnected from each other and thus can’t align.

With flexbox, they’re still disconnected, but their parents aren’t and can be equal height, plus...
New “auto” margin behavior

- Margins set to auto get all the free space left
- So, to pin flex item to bottom of its flex container:
  - set `flex-direction:column` on flex container so items can fill its height
  - set `margin-top: auto` on item you want to pin
Pin the “other” fields

1. Make each `.component` match in height by making parent `.builder` a flex container (already done)

```
.builder {
  display: flex;
  align-items: stretch; /* default */
  flex-wrap: wrap;
  justify-content: space-between;
  margin: 0 0 40px -20px;
}
```
Pin the “other” fields

2. Make each `<ul>` a flex item and stretch to full height, then make it a flex container with vertical direction so its `<li>` will stack

```
.component {
    flex: 1 1 200px;
    display: flex;
    flex-direction: column;
}

.component ul {
    flex: 1;
    display: flex;
    flex-direction: column;
    margin: 0;
    padding: 0;
    list-style: none;
}
```
Pin the “other” fields

3. Give “other” `<li>` an auto top margin so all free space left in `<ul>` is put above that `<li>`, pushing it to bottom of `<ul>`

```
.component li:last-child {
    margin-top: auto;
}
```
Pinning without flexbox

- Use `display: table-cell` for equal height boxes
- Add bottom padding in ems to each box
- Use absolute positioning to pin “other” row in space left by padding
Variation: two-piece main nav

.flexbox .list-nav {
    justify-content: flex-start;
    position: relative;
    top: -70px;
}
.flexbox #link-home { margin-right:20px; }
.flexbox #link-tumblr { margin-left:20px; }
.flexbox #link-party {
    margin-left: auto;
}
Why go to the trouble to use flexbox as progressive enhancement now?"
Develop it as a career skill

- Essential layout tool in the future, especially with responsive web design
- Syntax is not going to change much, so what you learn now will still work later
- Better to learn something before it’s needed in a project rather than during
"Why should I do it with CSS when I can do the same thing with `<font>` tags?"

– Zoe, circa 2002
Why should I do it with flexbox when I can do the same thing with floats?"
We all learn best by doing

- I learned a lot more about flexbox by building demo site for this presentation—a lot
- Have to try it to learn it
- Using it for small cosmetic enhancements is low-risk way to try it
It’s fun

- Great user experience is important, but great developer experience is worthwhile too
- Enjoy your job to get better at your job
- Sometimes the little extra time is worth the fun challenge and reward at the end
Learn more

Download slides and get links at www.zomigi.com/blog/flexbox-presentation

Thanks!
Zoe Mickley Gillenwater
@zomigi
design@zomigi.com
zomigi.com | stunningcss3.com | flexiblewebbook.com