# IS FLEXBOX THE FUTURE OF LAYOUT?

**james williamson | lynda.com**

CSS Dev Conference

# Hello.

**I’m James Williamson**

lynda.com | senior author

happy to be here

**@jameswillweb on the Twitter**

CSS Dev Conference

# What are we going to talk about?

**Flexbox… and can it fix what's wrong with CSS layout?**

CSS Dev Conference

# So wait... what's wrong with layout now?

**Let's take a look at the evolution of CSS layout:**

## no layout

We just let normal flow do its thing

## tables

For you youngsters.... seriously, we used to do this

## box model / positioning / floats / cursing

Progress! Still, much like tables, we're still hacking...

# Things that should be simple aren't

**Be honest. How many techniques can you think of, just off the top of your head, for vertically centering something?**

How easy are they?

I rest my case.

# Which brings us to Flexbox

> In the flex layout model, the children of a flex container can be laid out in any direction, and can "flex" their sizes, either growing to fill unused space or shrinking to avoid overflowing the parent. Both horizontal and vertical alignment of the children can be easily manipulated. Nesting of these boxes (horizontal inside vertical, or vertical inside horizontal) can be used to build layouts in two dimensions.
>
> — CSS Flexible Box Layout Module

# What's so great about Flexbox?

**Some of the high points:**

### It's flexible!

I mean, look at the name. You can easily make things stretch and flex to fit available space

### Easy alignment

Horizontal, vertical, baseline… it's all good.

### Source order independence

You want that before this, but only when that? OK.

### Easy Syntax

You can learn it in one afternoon.

Sounds awesome! So...is Flexbox the future of layout?

**No.**

Thanks for coming.

CSS Dev Conference

# Sounds awesome! So...is Flexbox the future of layout?

**OK, just kidding (sort of). The answer is both Yes & No!**

CSS Dev Conference

# First, a little history...

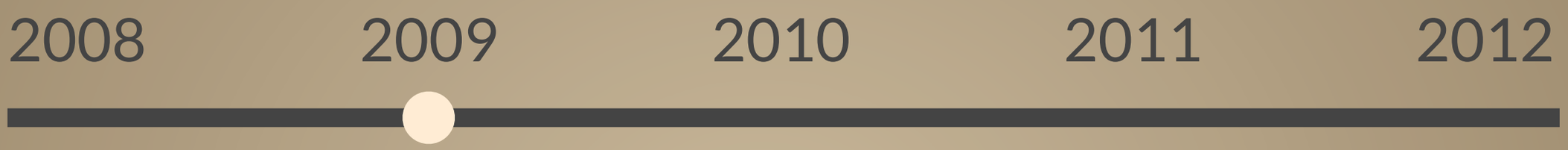
## pre 2008

CSS Working Group discusses proposing a Flexible Box Model similar to what is found in XUL and XAML.

# First, a little history...

## 2009

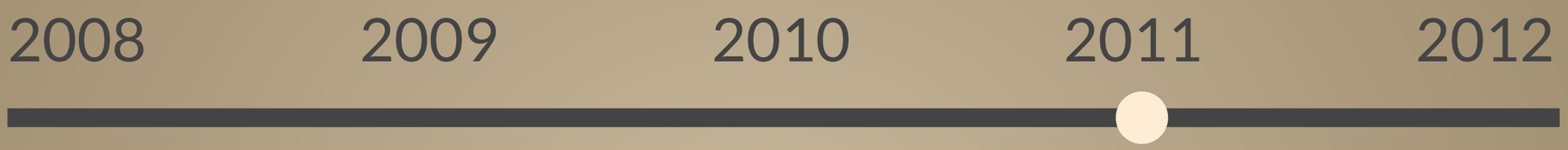
The Flexible Box Layout Module is published as a working draft. Chrome and Safari add partial support while Mozilla support relies on older XUL implementation. The syntax closely follows XUL flexbox syntax and is often referred to as “Flexbox 2009.”



# First, a little history...

## 2011

Tab Atkins takes over as editor for the Flexbox Spec and publishes two working drafts over the course of the year. These drafts re-write the syntax significantly and are sometimes referred to as “tweener” syntax. Chrome, Opera and & IE 10 begin to implement this syntax.



# First, a little history...

## 2012

Syntax is further changed and refined. Spec is now a Candidate Recommendation. Opera releases un-prefixed support, Chrome supports the current syntax in prefixed form, and IE 10 adds prefixed support for the “tweener” syntax. Mozilla is close to releasing unprefixed support.

2008 2009 2010 2011 2012

# How does Flexbox work?

**Basic concepts:**

It's a new layout mode

Joins block, inline, table, and positioned

Similar to block layout

Containing elements are laid out in a flow direction

Has super powers

Flow direction can be up or down, left or right, display order can be reversed, elements can "flex" their size to respond to available space and align to their containers or other elements

# How does Flexbox work?

**Basic steps:**

## Define flex containers

All direct child elements become flex items

## Establish flow direction

Flex containers can flow either in a row or column and can be single or multiline

## Go crazy with the cheese whiz

Flex items can now be spaced, flexed, aligned, or arranged as you see fit within the flow direction

# How does Flexbox work?

**It's all about the axis… er axes.**


CSS Dev Conference

# Flexbox syntax*

## Defining flex containers

```
.flex {
   display: -webkit-flex;
   display: -ms-flexbox;
   display: flex;
}
```

(there's also an **inline-flex** variation)

**A note about syntax. I'll be showing 2012 syntax including webkit prefixes, IE prefix ("tweener" syntax), and unprefixed syntax. I won't show older (2009 syntax) or older -o- and -moz- prefixes*

# Flexbox syntax

## Setting flow direction and line wrap

```
.flex {
   -webkit-flex-flow: <flex direction> | <flex wrap>
   -ms-flex-flow: “”
   flex-flow: “”
}
```

You can also set **flex-direction** and **flex-wrap** as individual properties if you wish.

# Flexbox syntax

## Controlling flex item flexibility

```
.flex-item {
   -webkit-flex: none |<flex-grow> <flex-shrink>|| <flex-basis>
   -ms-flex: “”
   flex: “”
}
```

I’m not going to lie… this takes some explaining…

# Flexbox syntax

## **Flex: Flex Grow**

```
.flex-item {
   -webkit-flex: none |<flex-grow> <flex-shrink>|| <flex-basis>
   -ms-flex: “”
   flex: “”
}
```

**<number>** Represents how much the flex item will grow relative to the rest of the flex items in the container once positive space has been distributed. If left out, it defaults to ‘**1**’

# Flexbox syntax

## Flex: Flex Shrink

```
.flex-item {
   -webkit-flex: none |<flex-grow> <flex-shrink>|| <flex-basis>
   -ms-flex: “”
   flex: “”
}
```

**<number>** Represents how much the flex item will shrink relative to the rest of the flex items in the container once negative space has been distributed. If left out, it defaults to ‘**1**’

# Flexbox syntax

## Flex: Flex Basis

```
.flex-item {
   -webkit-flex: none |<flex-grow> <flex-shrink>|| <flex-basis>
   -ms-flex: “”
   flex: “”
}
```

**auto | <width>** Represents the initial main size of a flex item, before free space is distributed. When omitted, it defaults to ‘0’

# Flexbox syntax

## Common Flex Values

### Flex: 0 auto, initial

Equates to **0 1 auto**. Sizes items based on width/height values. Item is inflexible but is allowed to shrink to its min value

### Flex: auto

Equates to **1 1 auto**. Sizes items based on width/height values, but makes them fully flexible to grow or shrink based on available space

### Flex: none

Equates to **0 0 auto**. Sizes items based on width/height values, but makes the item totally inflexible.

### Flex: <positive number>

Equates to **<value> 1 0px**. Makes the item flexible and sets the basis to 0. This ensures the item receives the specified portion of free space available.

# Flexbox syntax

## Controlling display order

```
.flex-item {
   -webkit-order: <integer>
   -ms-flex-order: “”
   order: “”
}
```

Values start at ‘0’ and increments up. A negative value is displayed before positive values. You can also reverse row and column direction.

# Flexbox syntax

## Controlling main axis alignment

```
.flex {
-webkit-justify-content: flex-start | flex-end | center |
                         space-between | space-around
-ms-flex-pack: start | end | center | justify
justify-content:””
}
```

Axis alignment is performed after flexible lengths and auto margins have been resolved.

# Flexbox syntax

## Controlling cross axis alignment

```
.flex {
-webkit-align-items: flex-start | flex-end | center |
                     baseline | stretch
-ms-flex-align: start | end | center | baseline | stretch
justify-content:””
}
```

Align-items applies to all flex items in a container. To align a single item, you can use the **align-self** property to a flex item and use the same values.

# Flexbox syntax

## Aligning multiple flex lines

```
.flex {
-webkit-align-content: flex-start | flex-end | center |
                       space-between | space-around | stretch
-ms-flex-line-pack: start | end | center | justify | distribute
                    | stretch
align-content:”” }
```

Aligns multiple flex lines within a flex container. Has no effect on single line flex containers.

# Fantastic


Let’s take it for a spin.

CSS Dev Conference

# Demo Time


If you're viewing this slide deck later,
this is where the cool stuff happened.
Sorry.

CSS Dev Conference

# Demo take-aways

**Some things to remember:**

### Don’t overuse it

Let normal flow do the work where it makes sense

### Think through your structure carefully

Defining regions and re-ordering content properly does rely on structure, think these things through

### Understand flex-basis

Knowing how an element’s main and cross size’s are determined is crucial to achieving expected results

### Don’t forget your margins

When setting alignments along axes, margins are taken into account. Also, flex item margins don’t collapse.

# So... how's support?

| Browser | support | notes |
|---|---|---|
| Firefox | ! 2.0+ | new syntax unprefixed in nightly |
| Google Chrome | + 22+ | with -webkit- prefix |
| Safari | ! 5.1+ | |
| Opera | + 12.1+ | |
| Internet Explorer | ! 10+ | -ms- prefix, uses tweener syntax |
| iOS Safari | ! 3.2 | 2009 syntax |
| Opera Mini | − 5 - 7 | |
| Opera Mobile | + 12.0+ | |
| Android Browser | ! 2.1 | 2009 syntax, -webkit- prefix |
| Blackberry Browser | + 10+ | Wait.... what? |

+ 2012 syntax supported

! 2009 syntax

− nada

*(as of 12-05-12)*

# So wait… is it the future or not?

**Of course… along with other emerging models**

## It's great at 1D, OK at 2D

This makes Flexbox a great choice for UI elements, application interfaces, and aligning/flexing items in specific page regions

## It's not great at 3D or across page regions

CSS Grid Layout is a better choice for that

## So what will we probably see?

Eventually I see Flexbox being used in conjunction with other layout models to exact finer-grain control over responsive elements

# Go learn you some Flexbox

Go read the spec:

http://www.w3.org/TR/css3-flexbox/

Browser support:

http://caniuse.com/flexbox

Using Flexbox:

https://developer.mozilla.org/en-US/docs/CSS/Using_CSS_flexible_boxes

Layout Nirvana?

http://dev.opera.com/articles/view/flexbox-basics/

Flexbox Please!

http://demo.agektmr.com/flexbox/

Want these slides?

http://www.slideshare.net/jameswillweb/presentations

CSS Dev Conference

# THANK YOU

**james williamson | lynda.com**

**jwilliamson@lynda.com**

**@jameswillweb on the Twitter**

**www.simpleprimate.com**

CSS Dev Conference