LAYOUT THE WEB WITH CSS GRID

and the rest of team layout

#TeamWeb

Original image by the amazing Lin Clark
This workshop has been inadvertently brought to you by Philippine Airlines.
Surname: Chen
First name: Hui Jing
Username: @hj_chen
Starter files

Evolution of browsers

- No layout
- HTML Tables
- CSS Floats
- Frameworks
- Grid and beyond
```html
<!doctype html>
<html lang="en">
  <head>
    <title>CSS rocks</title>
  </head>
  <body>
    <div>
      <h1>Title</h1>
      <p>Flying rabbits, whoa</p>
    </div>
    ...
    ...
  </body>
</html>
```
Box dimensions and type

Positioning scheme
(normal flow / float / absolute positioning)

Layout of boxes

Relationships between elements in the document tree

External information
(e.g. viewport size, intrinsic dimensions of images etc.)
Positioning schemes

Normal flow

Floats

Absolute positioning
Basic terminology

Grid line

Grid track

Grid area

Grid cell

Grid gap
The container–child relationship

Flex/Grid container

Flex/Grid item
“Grid works from the container in, other layout methods start with the item”

—Rachel Andrew
### Layout technique: inline-block

<table>
<thead>
<tr>
<th>Item A</th>
<th>Item B</th>
<th>Item C</th>
</tr>
</thead>
<tbody>
<tr>
<td>Item D</td>
<td>Item E</td>
<td>Item F</td>
</tr>
</tbody>
</table>

```css
class inlineblock_item {
    width: calc(100% / 3);
    display: inline-block;
}
```
### Layout technique: float

<table>
<thead>
<tr>
<th>Item A</th>
<th>Item B</th>
<th>Item C</th>
</tr>
</thead>
<tbody>
<tr>
<td>Item D</td>
<td>Item E</td>
<td>Item F</td>
</tr>
</tbody>
</table>

CSS code for float layout:

```
.float_item {
  width: calc(100% / 3);
  float: left;
}
```
Layout technique: `flex`
“Grid is the only layout technique that establishes a relationship between rows and columns of grid items.”
Firefox Grid Inspector
THE MOST BASIC GRID
Defining a grid

Using `grid-template-rows` and `grid-template-columns`
AUTO-PLACEMENT OF GRID ITEMS
(and the implicit grid)
The `repeat()` function

To specify a large number of columns or rows that follow a similar pattern

<table>
<thead>
<tr>
<th>Item</th>
<th>Item</th>
<th>Item</th>
<th>Item</th>
<th>Item</th>
<th>Item</th>
<th>Item</th>
<th>Item</th>
</tr>
</thead>
</table>

```css
.grid2 {
  display: grid;
  grid-template-columns: repeat(4, 75px 120px);
}
```
**auto-fill versus auto-fit**

Letting the browser figure out the math

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
</tr>
</thead>
</table>

```css
.keyword {
  display: grid;
  grid-template-columns: repeat(auto-fill, minmax(100px, 1fr));
}
```
grid-auto-row and grid-auto-column

Controlling the size of implicit rows/columns

```
.grid3 {
    display: grid;
    grid-template-columns: 150px 150px 150px;
    grid-template-rows: 150px;
    grid-auto-rows: 120px;
}
```
The **grid-auto-flow property**

Adjusting the direction and density of grid items

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>E</td>
<td></td>
<td>F</td>
<td>I</td>
</tr>
<tr>
<td></td>
<td>G</td>
<td></td>
<td>L</td>
</tr>
</tbody>
</table>

```css
.grid4 {
  display: grid;
  grid-template-columns: repeat(auto-fit, minmax(120px, 1fr));
  grid-auto-rows: 120px;
  grid-auto-flow: dense row;
}
.grid4__item:nth-child(3n+1) {
  grid-column-end: span 2;
}
.grid4__item:nth-child(2n+3) {
  grid-row-end: span 2;
}
.grid4__item:nth-child(3n+5) {
  grid-column-end: span 3;
}
```
MORE WAYS TO SIZE GRID ROWS AND COLUMNS
The \textbf{fr} unit

Represents a \textit{fraction} of the \textbf{free space} in the grid container.

<table>
<thead>
<tr>
<th>Item A</th>
<th>Item B</th>
<th>Item C</th>
</tr>
</thead>
</table>

```css
.grid5 {
  display: grid;
  grid-template-columns: 150px 1fr 2fr;
}
```
Fluid CSS grid

.container {
  display: grid;
  grid-template-columns: repeat(3, 3fr 2fr);
}

CSS Grids
Fluid Grid
The **minmax()** function

Defines a size range for columns or rows in the grid.

```
.grid6 {
  display: grid;
  grid-template-columns:
  minmax(200px, 1fr) 300px 300px;
}
```
Responsive grid without media queries

```
.container {
  display: grid;
  grid-template-columns: repeat(auto-fill, minmax(10em, 1fr));
}
```
## Content-based sizing

Using the `min-content`, `max-content` and `fit-content` properties

<table>
<thead>
<tr>
<th>What's worse, lookin' jealous or crazy?</th>
<th>Or like being walked all over lately</th>
<th>I'd rather be crazy</th>
</tr>
</thead>
</table>

```css
.grid7 {
  display: grid;
  grid-template-columns: min-content fit-content(25ch) max-content;
}
```
Simple responsive grid
MANUAL PLACEMENT OF GRID ITEMS
(and the explicit grid)
Line-based placement

Using grid-row-start, grid-row-end and grid-column-start, grid-column-end

```css
.grid8 {  
display: grid;  
grid-template-columns: 33% 33% 33%;  
grid-template-rows: 20vh 20vh 20vh;  
}
.grid8__item {  
grid-column-start: 2;  
grid-column-end: 3;  
grid-row-start: 2;  
grid-row-end: 3;  
}
```
SOME GRID SHORTHANDS

(because brevity is a virtue...maybe)

\_\_(ツ)\_/\_
Using the `grid-row` and `grid-column` shorthands

By default, grid items will take up the space of 1 grid cell

```css
.grids9 {
  display: grid;
  grid-template-columns: 2em 2em 2em 2em 2em;
  grid-template-rows: 2em 2em 2em 2em 2em;
}

.grids9___item:nth-child(1) {
  grid-row: 2;
  grid-column: 5;
}

.grids9___item:nth-child(2) {
  grid-row: 4;
  grid-column: 5;
}

.grids9___item:nth-child(3) {
```
The `span` keyword

Content can span multiple grid cells

```
.grid10 {
    display: grid;
    grid-template-columns: 20% 20% 20% 20%;
    grid-template-rows: 12vh 12vh 12vh 12vh;
}
.grid10__item {
    grid-row: 1 / span 3;
    grid-column: 2 / span 2;
}
```
Using the `grid-area` shorthand

```
.grid-area: <grid-row-start> / <grid-column-start> / <grid-row-end> / <grid-column-end>
```

```
.grid11 {
  display: grid;
  grid-template-columns: 20% 20% 20% 20%;
  grid-template-rows: 20vh 20vh 20vh;
}
.grid11__item {
  grid-area: 2 / 2 / 4 / 6;
}
```
ASSIGNING NAMES TO GRID THINGS
Naming grid lines

```
.grid12 {
    display: grid;
    grid-template-columns: [alpha-start] 150px [alpha-end]
                        [beta-start] 150px [beta-end]
                        [gamma-start] 150px [gamma-end];
}
.grid12 .c {
    grid-column: alpha-start / beta-end;
}
```
Naming grid areas

Using `grid-template-areas` and `grid-area`
MAKING SENSE OF BOX ALIGNMENT
<table>
<thead>
<tr>
<th>Property</th>
<th>Axis</th>
<th>Aligns</th>
<th>Applies to</th>
</tr>
</thead>
<tbody>
<tr>
<td>justify-content</td>
<td>main/inlin</td>
<td>content within element (effectively adjusts padding)</td>
<td>block containers, flex containers and grid containers</td>
</tr>
<tr>
<td>align-content</td>
<td>cross/block</td>
<td></td>
<td></td>
</tr>
<tr>
<td>justify-self</td>
<td>inline</td>
<td>element within parent (effectively adjusts margins)</td>
<td>block-level boxes, absolutely-positioned boxes and grid items</td>
</tr>
<tr>
<td>align-self</td>
<td>cross/block</td>
<td></td>
<td>absolutely-positioned boxes, flex items and grid items</td>
</tr>
<tr>
<td>justify-items</td>
<td>inline</td>
<td>items inside box (controls child items)</td>
<td>block containers and grid containers</td>
</tr>
<tr>
<td>align-items</td>
<td>cross/block</td>
<td></td>
<td>flex-containers and grid-containers</td>
</tr>
</tbody>
</table>

Source: CSS Box Alignment Module Level 3
justify/align-content

content-distribution properties

```css
.content {
  justify-content: normal;
  align-content: normal;

  display: grid;
  grid-template-columns: repeat(3, 25%);
  grid-template-rows: repeat(3, 20%);
  grid-gap: 1em;
  grid-template-areas:
    "a b b"
    "a b b"
    "c c d";
}

.content_itemA {
  grid-area: a;
}
```
<table>
<thead>
<tr>
<th>Values</th>
<th>justify-content</th>
<th>align-content</th>
</tr>
</thead>
<tbody>
<tr>
<td>center</td>
<td><img src="image" alt="center" /></td>
<td><img src="image" alt="center" /></td>
</tr>
<tr>
<td>start</td>
<td><img src="image" alt="start" /></td>
<td><img src="image" alt="start" /></td>
</tr>
<tr>
<td>end</td>
<td><img src="image" alt="end" /></td>
<td><img src="image" alt="end" /></td>
</tr>
<tr>
<td>space-around</td>
<td><img src="image" alt="space-around" /></td>
<td><img src="image" alt="space-around" /></td>
</tr>
<tr>
<td>space-between</td>
<td><img src="image" alt="space-between" /></td>
<td><img src="image" alt="space-between" /></td>
</tr>
<tr>
<td>space-evenly</td>
<td><img src="image" alt="space-evenly" /></td>
<td><img src="image" alt="space-evenly" /></td>
</tr>
<tr>
<td>Values</td>
<td>justify-content</td>
<td>align-content</td>
</tr>
<tr>
<td>-------------------</td>
<td>-----------------</td>
<td>---------------</td>
</tr>
<tr>
<td>center</td>
<td></td>
<td></td>
</tr>
<tr>
<td>start</td>
<td></td>
<td></td>
</tr>
<tr>
<td>space-between</td>
<td></td>
<td></td>
</tr>
<tr>
<td>space-evenly</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
justify/align-self

self-alignment properties

```css
.self {
  display: grid;
  grid-template-columns: repeat(4, 1fr);
  grid-gap: lem;
  grid-auto-rows: calc(25% - lem);
  grid-template-areas:
    "a a b b"
    "a a b b"
    "c c d d"
    "c c d d";
}
.self_itemA {
  grid-area: a;
}
.self_itemB {
```

justify/align-items

defaults for justify/align-self

```css
.items {
  justify-items: normal;
  align-items: normal;

  display: grid;
  grid-template-columns: repeat(4, 1fr);
  grid-gap: 1em;
  grid-auto-rows: calc(25% - 1em);
  grid-template-areas:
    "a a b b"
    "a a b b"
    "c c d d"
    "c c d d";
}

.items_itemA {
  grid-area: a;
}
```
Simple responsive dashboard

I dunno... use your imagination

Okay, free to be a game board so let's centre the content of the board.
This is sort of the idea.
Also, let's centre the title, user and statistics. Then centre the controls both ways.
GRID VERSUS FLEXBOX?
GRID VERSUS FLEXBOX?

WRONG QUESTION
Grid AND Flexbox
FIGURING OUT FLEXBOX
Flex axes

Main axis

Cross axis

horizontal-tb row
Flex lines

nowrap

wrap

wrap-reverse
### Flex directions

<table>
<thead>
<tr>
<th>一</th>
<th>二</th>
<th>三</th>
<th>四</th>
<th>五</th>
<th>六</th>
<th>七</th>
</tr>
</thead>
<tbody>
<tr>
<td>8八</td>
<td>9九</td>
<td>10十</td>
<td>11十一</td>
<td>12十二</td>
<td>13十三</td>
<td>14十四</td>
</tr>
<tr>
<td>15十五</td>
<td>16十六</td>
<td>17十七</td>
<td>18十八</td>
<td>19十四</td>
<td>20二十</td>
<td></td>
</tr>
</tbody>
</table>

```css
.directions .wrapper {
  display: flex;
  flex-wrap: wrap;
  writing-mode: horizontal-tb;
  flex-direction: row;
}

.directions .box {
  height: 6em;
  width: 6em;
  border: 1px solid;
}
```
WHAT IS THE MOST COMMON MISTAKE DEVELOPERS MAKE WHEN USING FLEXBOX?
Not using the `flex` shorthand

“Authors are encouraged to control flexibility using the `flex shorthand` rather than with its longhand properties directly, as the shorthand `correctly resets` any unspecified components to accommodate common uses.”

—CSS Flexible Box Layout Module Level 1
### Basic `flex` keyword values

<table>
<thead>
<tr>
<th>initial</th>
<th>auto</th>
<th>&lt;positive-number&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 1 auto</td>
<td>1 1 auto</td>
<td>&lt;positive-number&gt; 1 0</td>
</tr>
<tr>
<td>cannot grow but can shrink when there isn't enough space</td>
<td>can grow and shrink to fit available space</td>
<td>can grow and shrink, extent of growth depends on flex factor</td>
</tr>
</tbody>
</table>
“When a box is a flex item, flex is consulted instead of the main size property to determine the main size of the box.”

—CSS Flexible Box Layout Module Level 1
The `flex` syntax

- Brackets are for grouping:
  
  ```
  [ <'flex-grow'> <'flex-shrink'>? || <'flex-basis'> ]
  ```

- Preceding type/word/group is optional.

- Separates 2 options, one or more must occur, order doesn't matter.
/* One value, unitless number: flex-grow */
flex: 3;

/* One value, width/height: flex-basis */
flex: 200px;
flex: 45em;

/* Two values: flex-grow | flex-basis */
flex: 1 25ch;

/* Two values: flex-grow | flex-shrink */
flex: 2 1;

/* Three values: flex-grow | flex-shrink | flex-basis */
flex: 2 3 30%;
Aligning with auto margins

```css
.automargin {
  display: flex;
}

.automargin div {
  border: 1px solid;
  margin: auto;
}
```
Defining “auto” by Elika Etemad (AKA fantasai)
Aligning along the main axis

justify-content helps distribute extra free space left over after flexible lengths and auto margins are resolved.

```css
.mainaxis .wrapper {
    display: flex;
    flex-wrap: wrap;
    justify-content: flex-end;
}

.mainaxis .box {
    height: 5em;
    width: 5em;
    border: 1px solid;
}
```
Aligning along the cross axis

align-items sets the default alignment for all flex items along the cross axis of the flex line. Over-ridable by align-self.
Packing flex lines

align-content aligns flex lines within the flex container if there is extra space along the cross-axis.
Responsive configuration page
IS IT SAFE TO USE CSS GRID IN PRODUCTION?
Falling back with style

Using @supports AKA feature queries

```css
.selector {
/* Styles that are supported in old browsers */
}

@supports (property:value) {
.selector {
/* Styles for browsers that support the specified property */
}
```
Diagonal header

the critical request
HOW WE CAN IMPROVE DESIGN USING GRIDS?

and

WHAT ARE SOME OF THE AMAZING TECHNIQUES YOU’VE SEEN THAT USE CSS GRID?
Overlap

The Orange Issue
Vertical whitespace

“Nowadays people are looking to move on from their current roles for growth and development rather than just a salary bump.”

“LinkedIn is more than just a job portal, use it as an avenue for networking and reaching out, as well as a way to share knowledge.”

“Be aware of what you want out of your career before going in for interviews, so that you won't find yourself in situations where there is obviously no synergy between yourself and the company in question.”

https://singaporecss.github.io/specials/s2701.html
CSS grid showcase

by Andy Barefoot

by Yuan Chuan

by Jon Kantner

by Andy Barefoot
CSS grid showcase

by Adrian Roworth

by Chen Hui Jing

by Aysha Anggraini

by Chen Hui Jing
Tycho

Tycho is an American ambient music project led by Scott Hansen as primary composer, songwriter, and producer. Hailing from San Francisco, California, he is known as ISO50 for his photographic and design works. His music is a combination of downtempo, vintage-style synthesizers and ambient melodies.
Q&A time!

(for the rest of the questions not already covered)

- What feature would you like to add to the current spec of CSS grid?
- Did you start off as a freelancer? How difficult was it to get your first client?
- Tips for when you feel inferior about your work?
- Describe design in three words?
“CSS is a team sport.”

—Me
Useful references

- CSS Grid Layout Module Level 1
- Codrops CSS Grid reference
- Grid by Example
- Learn CSS Grid
- Grid Auto-Placement Is Ready
- Automatizing the Grid
- Deep Dive into Grid Layout Placement
- CSS Grid Layout and positioned items
- CSS Logical Properties and Values in Chromium and WebKit
- Changes on CSS Grid Layout in percentages and indefinite height
- The Story of CSS Grid, from Its Creators
- CSS Grid Layout is Here to Stay
- The New Layout Standard For The Web: CSS Grid, Flexbox And Box Alignment
- What Happens When You Create A Flexbox Flex Container?
- Everything You Need To Know About Alignment In Flexbox
- Grid “fallbacks” and overrides
Salamat!

https://www.chenhuijing.com

@hj_chen

@huijing

Font used is Prospectus, by Dave Bailey