Many Flavours of Enterprise CSS Grid

By Chen Hui Jing / @hj_chen

Surname

First name

陈	慧 晶
Chen	Hui Jing

@hj_chen

shopify

Screens, screens, screens

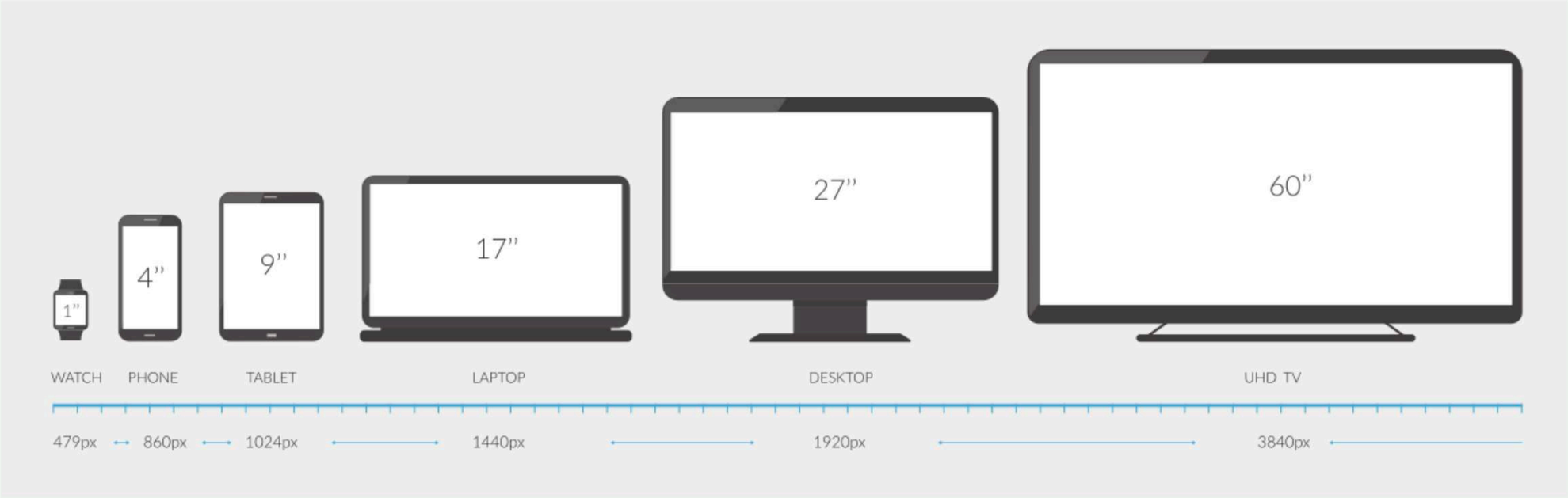


Image source: Inch Calculator

1.00

Image credit: Jyotika Sofia Lindqvist

```
.wrapper {
  display: -webkit-box;
  display: -webkit-flex;
  display: -ms-flexbox;
  display: flex; /* stable support by 2015 */
}
```

MDN: Backwards Compatibility of Flexbox

Grid release dates

March 2017

Sun	Mon	Tue	Wed	Thu	Fri	Sat
			1	2	3	4
5	6		8	9	10	11
12	13		15	16	17	18
19	20		22	23	24	25
26		28	29	30	31	

October 2017

Sun	Mon	Tue	Wed	Thu	Fri	Sat
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16		18	19	20	21
22	23	24	25	26	27	28
29	30					

CSS Flexible Box Layout Module

Method of positioning elements in horizontal or vertical stacks. Support includes all properties prefixed with `flex`, as well as `display: flex`, `display: inline-flex`, `align-content`, `align-items`, `align-self`, `justify-content` and `order`.

IE	Edge	Firefox	Chrome	Safari	iOS Safari	Opera Mini	Chrome for Android	Android Browser	Samsung Internet
9	103	102	103	15.5	15.5			4.4	16.0
10	104	103	104	15.6	15.6			4.4.4	17.0
11	105	104	105	16.0	16.0	all	105	105	18.0
		105	106	16.1	16.1				

✓ ✗ Partial Support Prefixed

Global: 98.71% + 0.99% = 99.70%

Data from caniuse.com | Embed from caniuse.bitsofco.de

Disable accessible colours

CSS Grid Layout (level 1)

Method of using a grid concept to lay out content, providing a mechanism for authors to divide available space for layout into columns and rows using a set of predictable sizing behaviors. Includes support for all `grid-*` properties and the `fr` unit.

IE	Edge	Firefox	Chrome	Safari	iOS Safari	Opera Mini	Chrome for Android	Android Browser	Samsung Internet
9	103	102	103	15.5	15.5			4.4	16.0
10	104	103	104	15.6	15.6			4.4.4	17.0
11	105	104	105	16.0	16.0	all	105	105	18.0
		105	106	16.1	16.1				

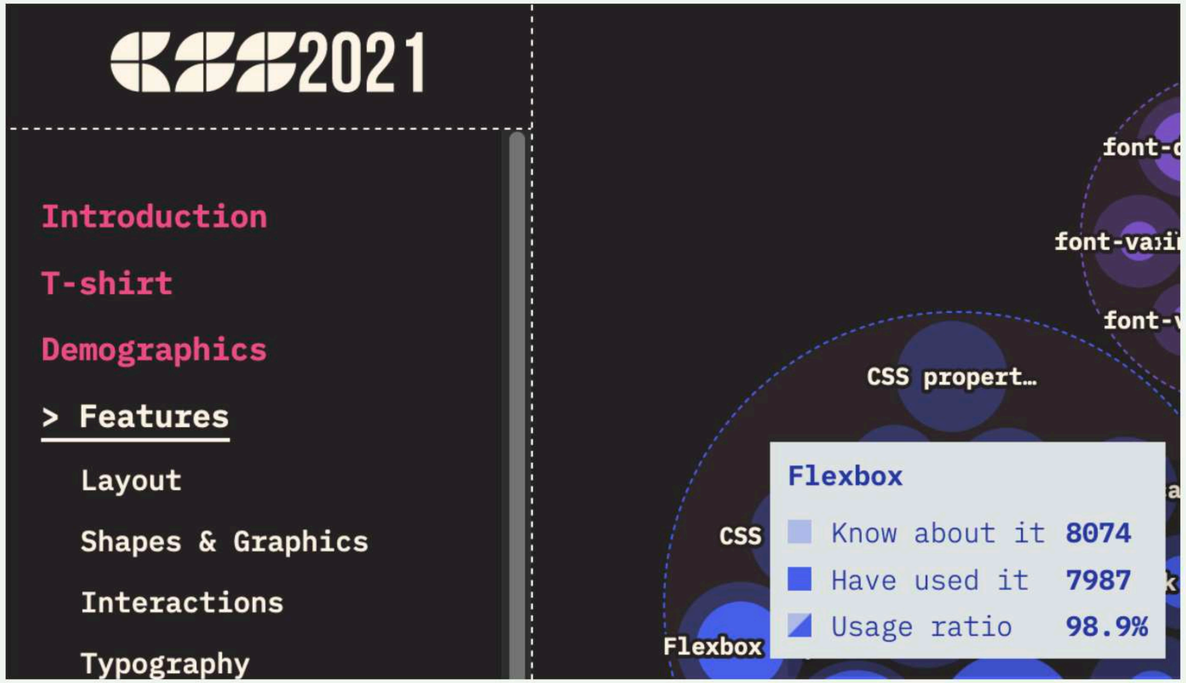
✓ ✗ Partial Support Prefixed

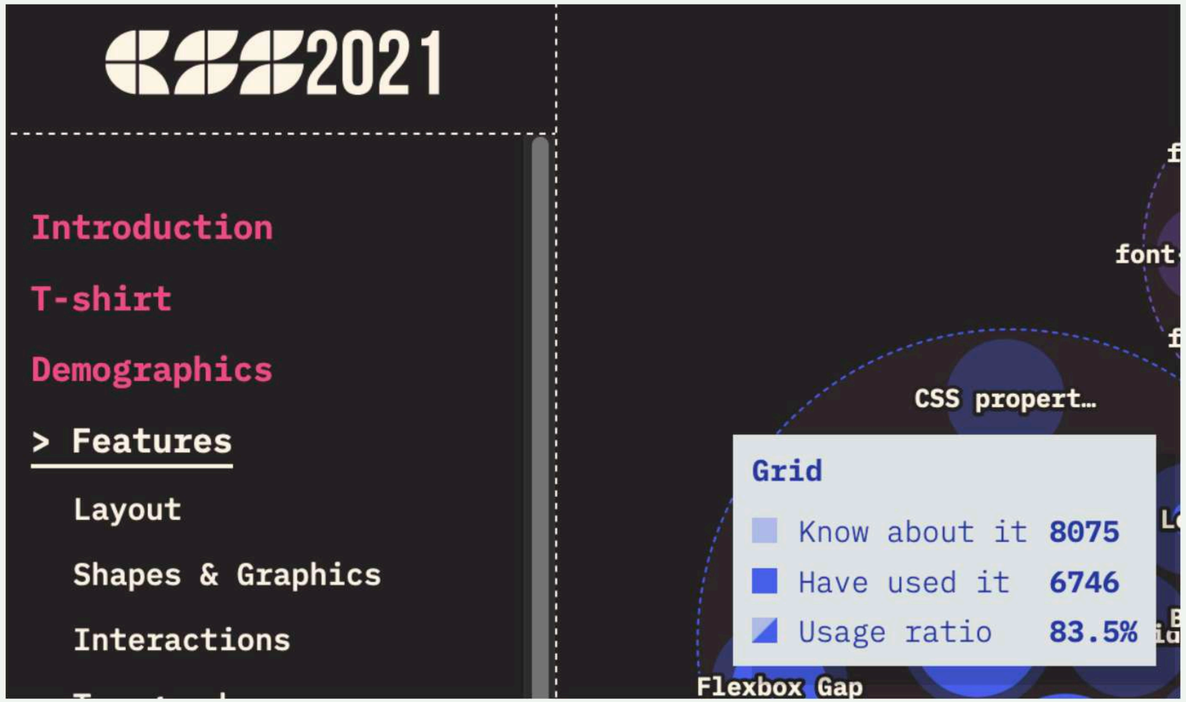
Global: 96.3% + 0.51% = 96.81%

Data from caniuse.com | Embed from caniuse.bitsofco.de

Disable accessible colours

State of CSS 2021 survey





Flexbox: 98.9%

Grid: 83.5%

https://2021.stateofcss.com/en-US/

Spotify

Upgrade

hj_ch

ome

earch

our Library

reate Playlist

ked Songs

ng Ji

nav-bar

ng

Sokko

s a day

ame Symphony

da Jam

z Cat

nd I

orite Coffeehouse

stall App

Good evening

RENAISSANCE

This Is CHVRCHES

Pop Rising

This Is girl in red

Today's Top Hits

Tones and I

Your top mixes

SE

main-view

Lizzo Mix
Beyoncé, Tinashe, Cardi B and more

Beyoncé Mix
Tems, Lizzo, City Girls and more

Hebe Tien Mix
Sam Lee, WeiBird, Jay Chou and more

2010s Mix
Tones And I, Lizzo, Hayley Kiyoko and more

80s Mix
a-ha, James Ingram, CoCo Lee and more

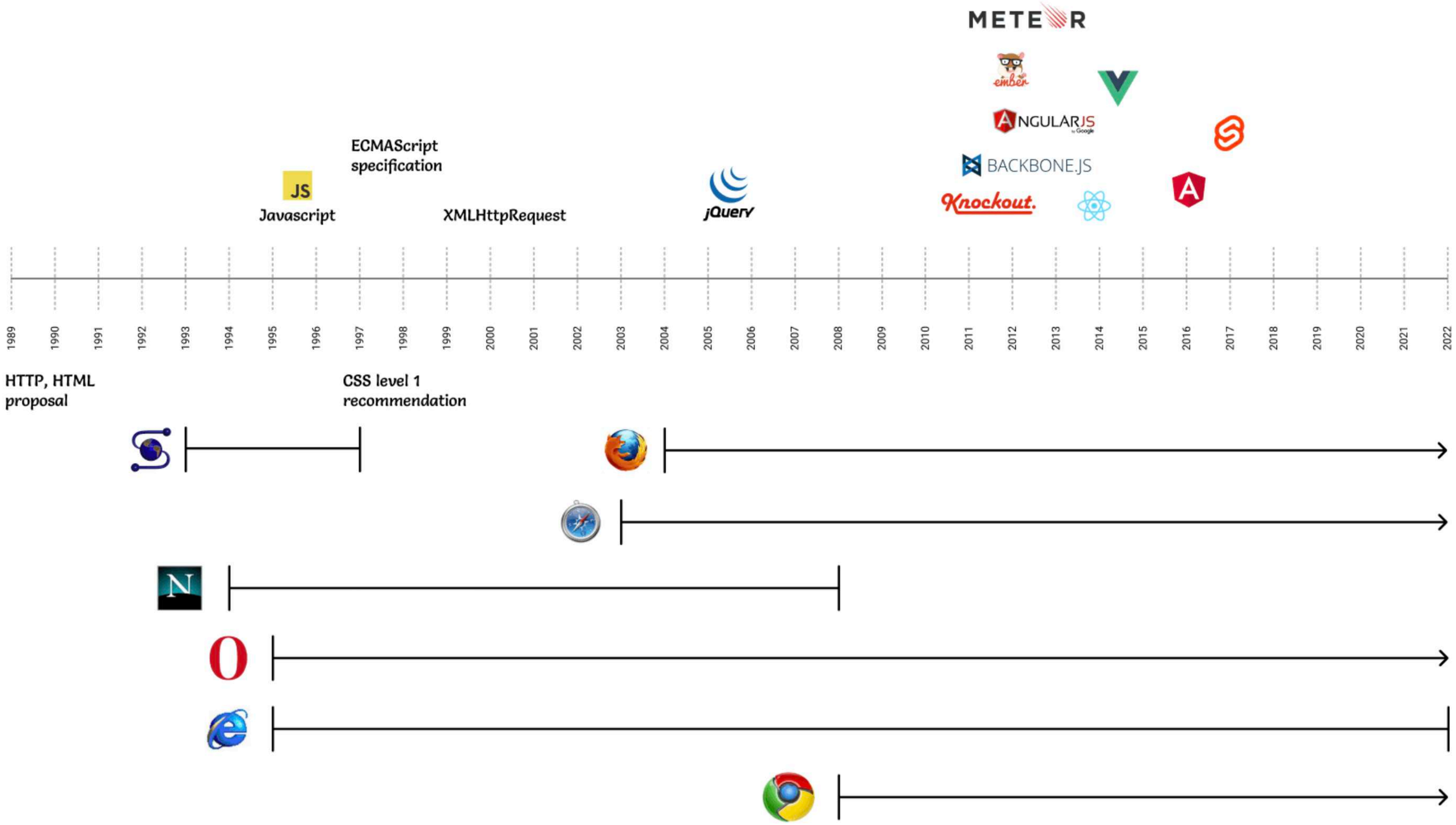
2000s Mix
Rene Liu, Sam Lee and more

Today's biggest hits

SE

Singapore

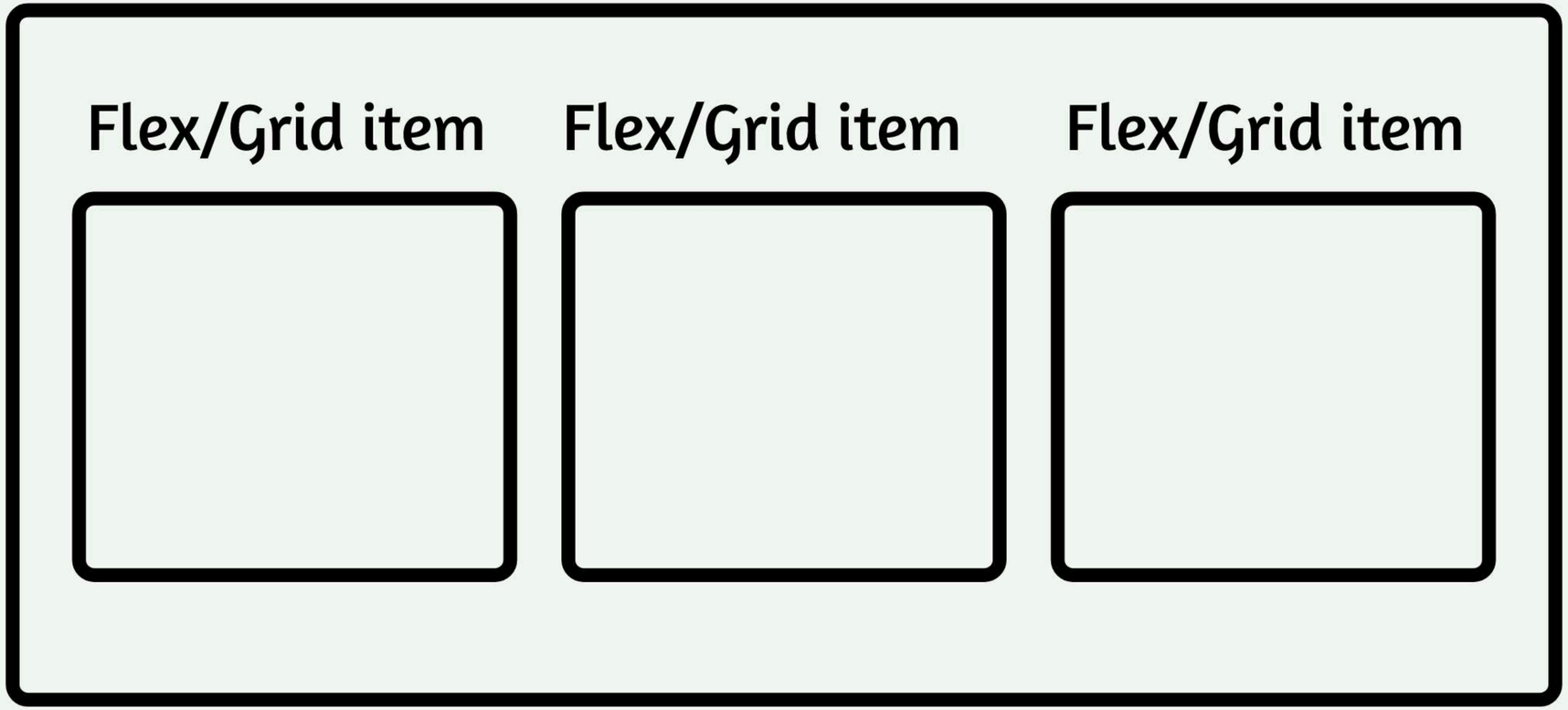
K-Pop


METEOR
ember
ANGULARJS
BACKBONE.JS
Knockout.
ECMAScript specification
JS
Javascript
XMLHttpRequest
jQuery
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
HTTP, HTML proposal
CSS level 1 recommendation

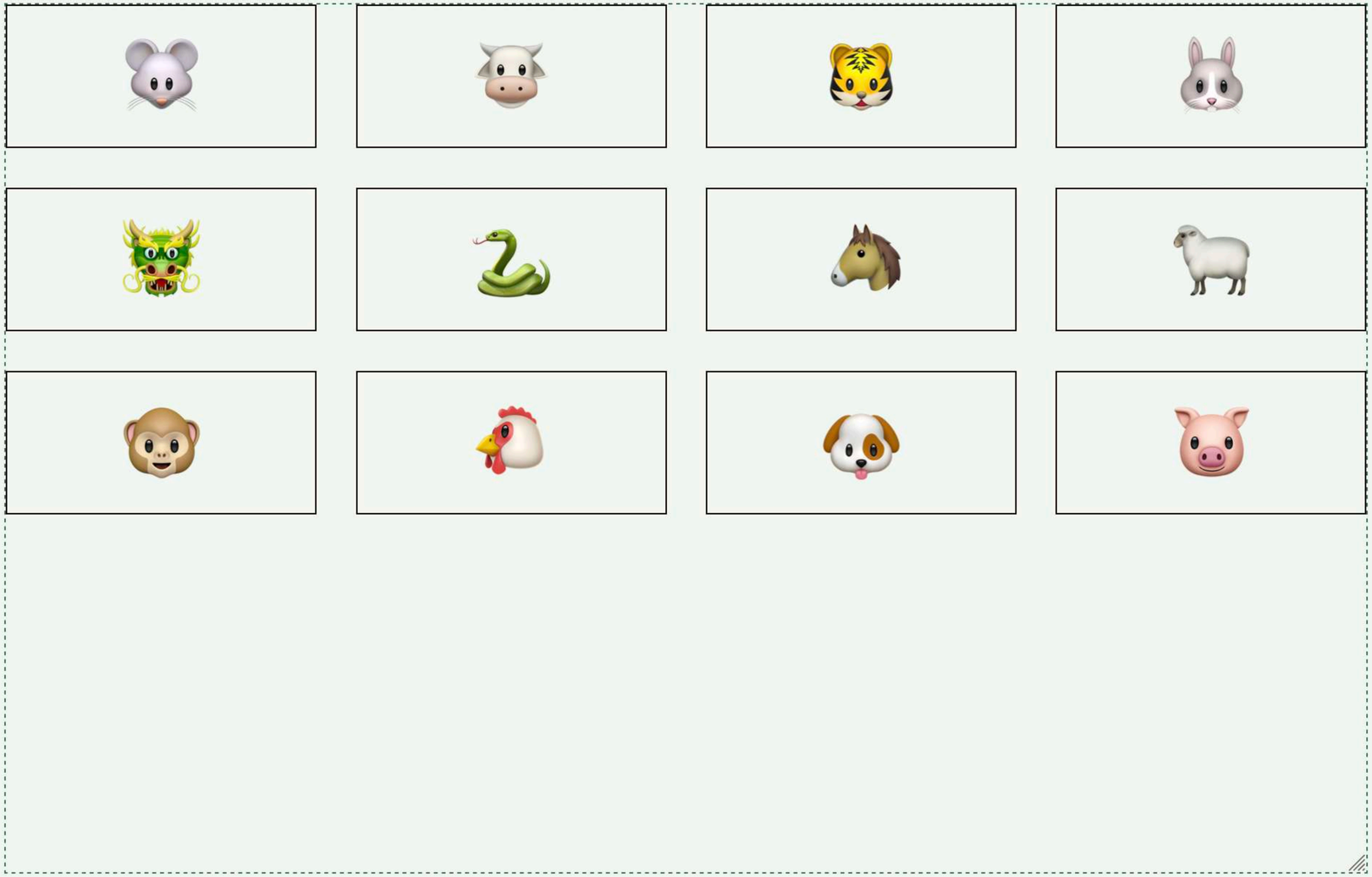
SPA

Parent-child relationship

Flex/Grid container

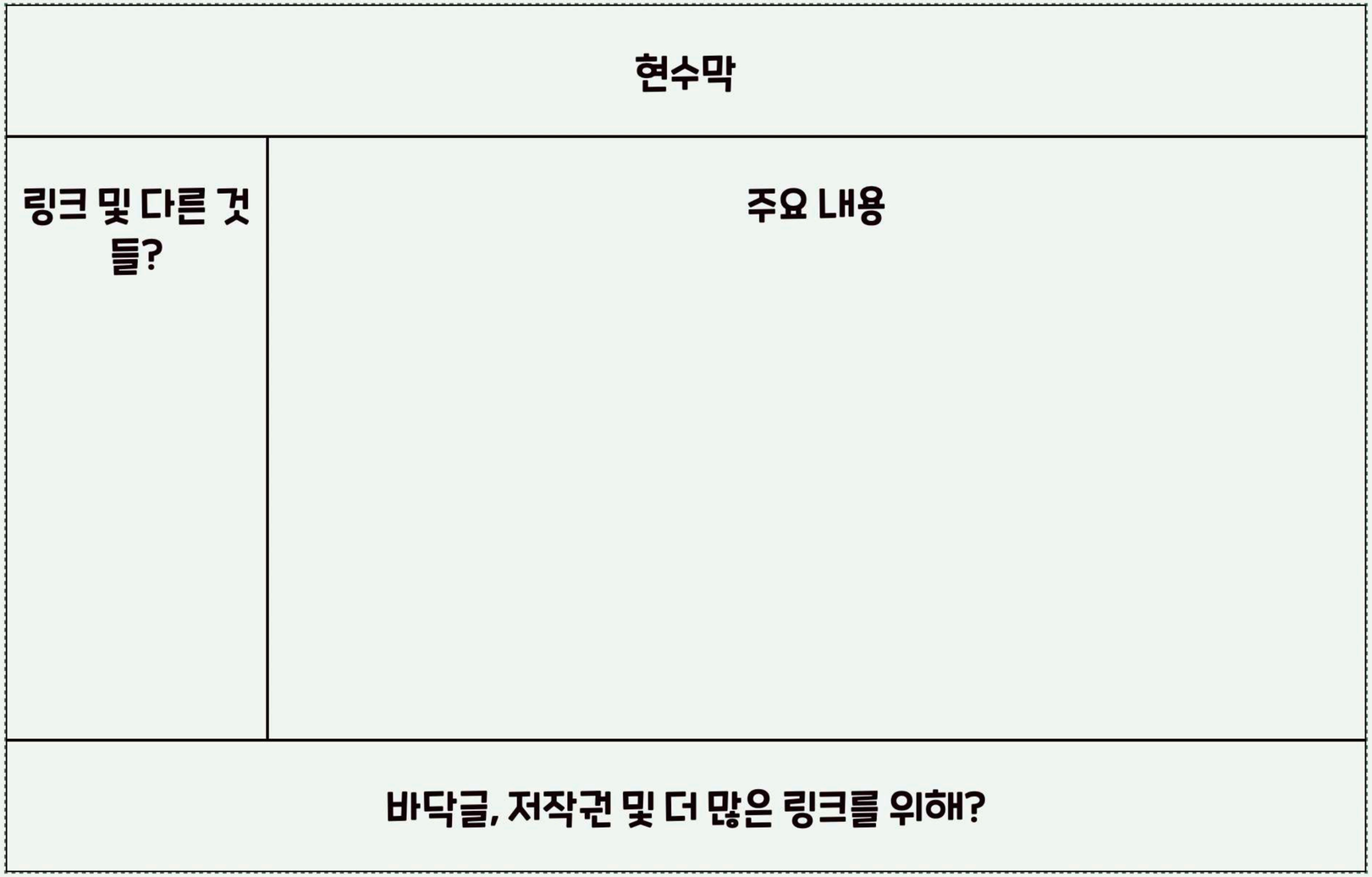


Basic grid syntax



```
.basic-grid {
  display: grid;
  grid-template-columns:
repeat(auto-fit, minmax(200px,
1fr));
  gap: 1em;
}
```

Named grid areas




```
.named-grid {
  display: grid;
  grid-template-columns: 200px 1fr;
  grid-template-rows: auto 1fr auto;
  grid-template-areas: '머리글 머리글'
                       '사이드 콘텐츠'
                       '바닥글 바닥글';
}

.h { grid-area: 머리글 }
.s { grid-area: 사이드 }
.m { grid-area: 콘텐츠 }
.f { grid-area: 바닥글 }
```

Placing grid items

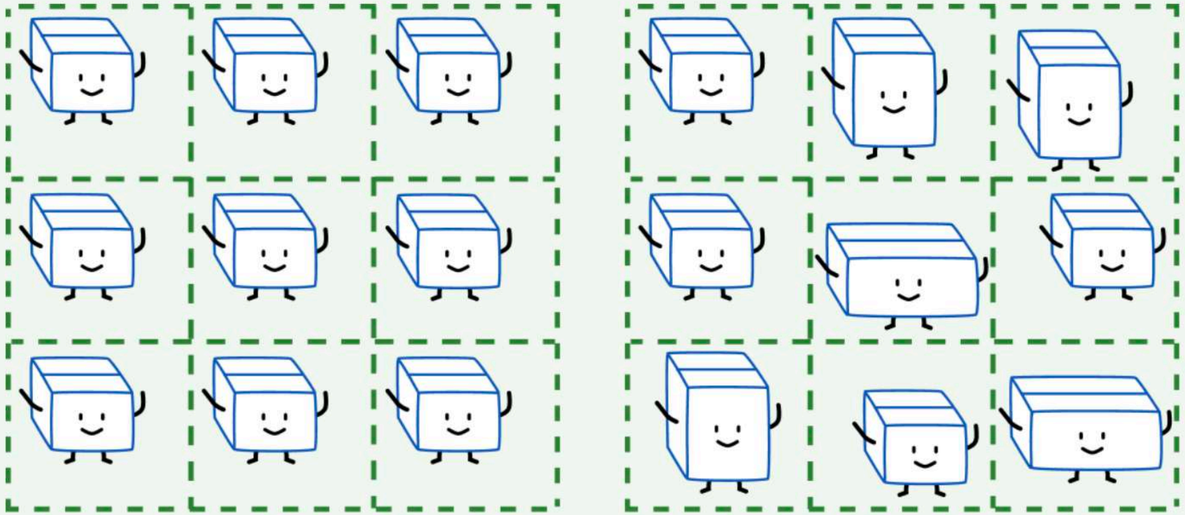
내 고양이가 아니다

그는 가끔 우리 집에서 낮잠을 잔다. 그런데 정말 귀엽지 않나요?

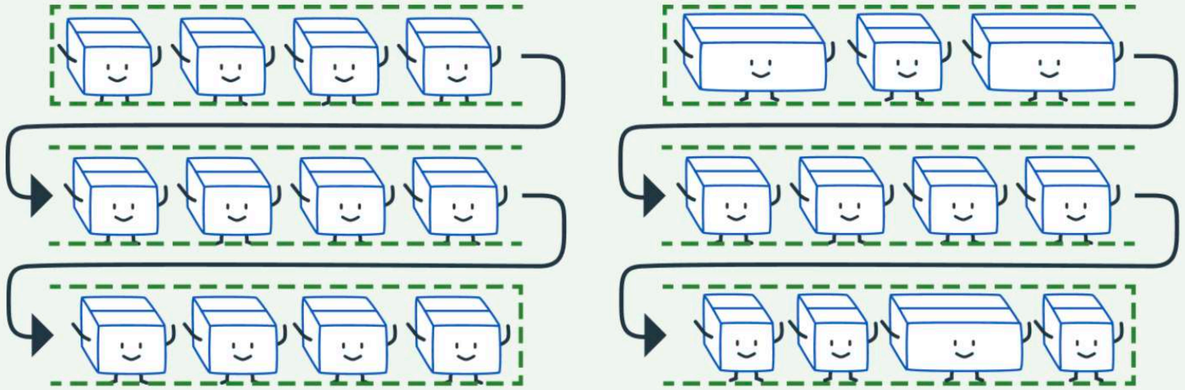
```
.overlap-grid {
  display: grid;
  grid-template-columns: 1fr 300px 1fr;
  grid-template-rows: 1fr auto 1fr;
}

.image {
  grid-column: 1 / span 2;
  grid-row: 1 / -1;
}
.headline {
  grid-column: 2 / 4;
  grid-row: 2;
}
.text {
  grid-column: 2 / 4;
  grid-row: 3;
}
```

Container-led sizing



Item-led sizing



Common example of grid system CSS

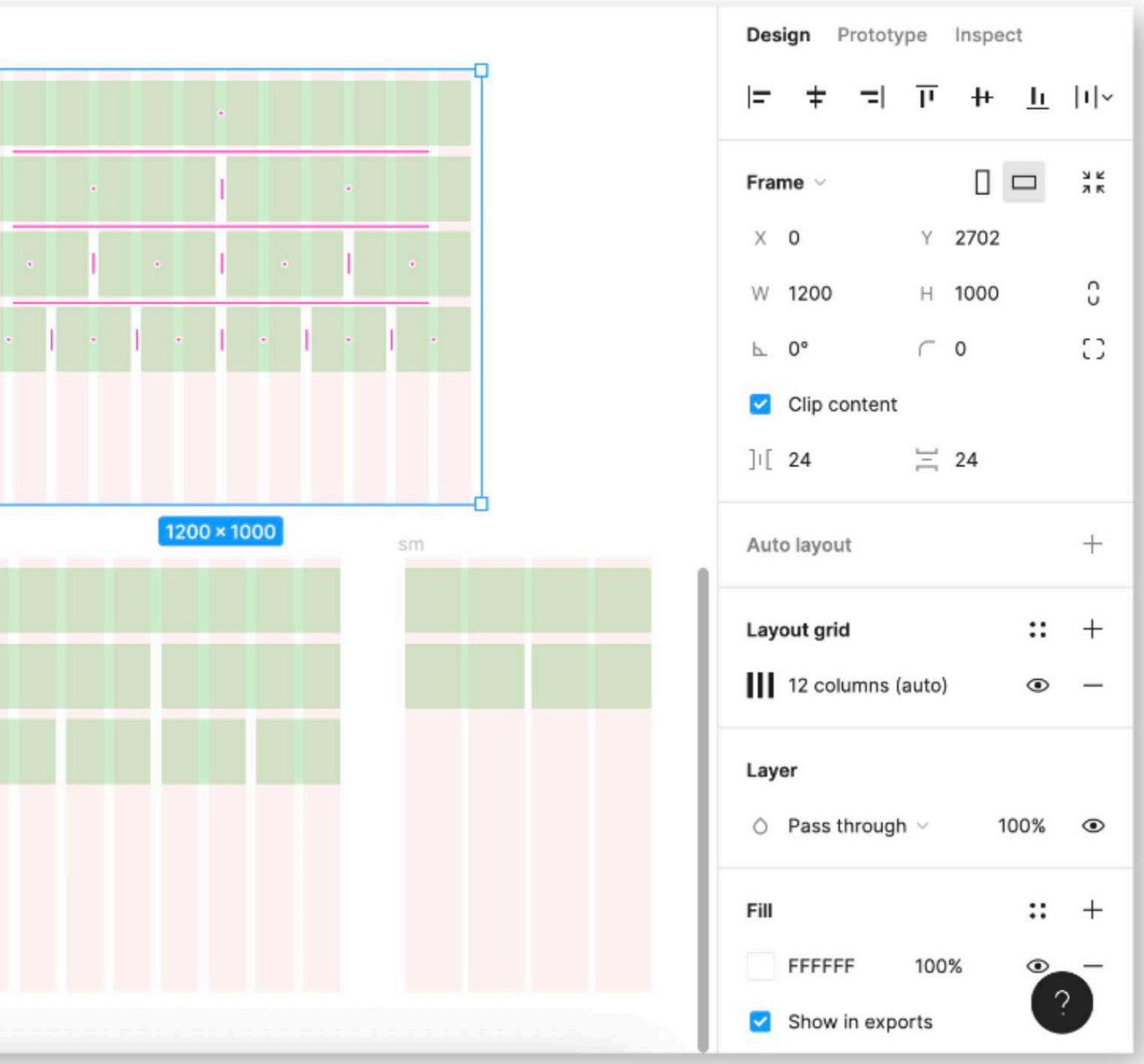
"Bootstrap's grid system uses a series of containers, rows, and columns to layout and align content."

```
.col-sm {
  flex: 1 0 0%;
}
.row-cols-sm-auto > * {
  flex: 0 0 auto;
  width: auto;
}
.row-cols-sm-1 > * {
  flex: 0 0 auto;
  width: 100%;
}
.row-cols-sm-2 > * {
  flex: 0 0 auto;
  width: 50%;
}
.row-cols-sm-3 > * {
```

```
<div class="container">
  <div class="row">
    <div class="col-sm-4">
      One of three columns
    </div>
    <div class="col-sm-4">
      One of three columns
    </div>
    <div class="col-sm-4">
      One of three columns
    </div>
  </div>
</div>
```

A pretty standard grid

Size	Min	Max	Cols	Margin	Gutter
xs	320px	639px	4	16px	16px
sm	640px	899px	8	30px	16px
md	900px	1199px	12	50px	16px
lg	1200px	1599px	12	90px	24px
xl	1600px	–	12	>180px	24px




```
import { ReactNode, createElement } from "react";
import styles from "./Grid.module.scss";

interface GridProps extends React.HTMLProps<HTMLElement>
  className?: string;
  children: ReactNode;
  tag?: keyof JSX.IntrinsicElements;
}

export default function Grid({
  className = "",
  children,
  tag = "div",
  ...props
}: GridProps) {
  const Wrapper = tag;
  return createElement(
    Wrapper,
    {
      className: `${styles.grid} ${className}`,
      ...props
    },
    children
  );
}
```

Option 1: vanilla CSS (or SCSS)

Size	Min	Max	Cols	Margin	Gutter
xs	320px	639px	4	16px	16px
sm	640px	899px	8	30px	16px
md	900px	1199px	12	50px	16px
lg	1200px	1599px	12	90px	24px
xl	1600px	–	12	>180px	24px

```
.grid {
  min-width: 320px;
  max-width: 1600px;
  display: grid;
  grid-template-columns: repeat(4, 1fr);
  gap: 1em;
  margin-left: 16px;
  margin-right: 16px;
}

@media screen and (min-width: 640px) {
  .grid {
    grid-template-columns: repeat(8, 1fr);
    margin-left: 30px;
    margin-right: 30px;
  }
```

Option 1: vanilla CSS (or SCSS)

```
.grid__item--full,
.grid__item--half,
.grid__item--third,
.grid__item--quarter {
  grid-column: 1 / -1;
}

@media screen and (min-width: 640px) {
  .grid__item--quarter {
    grid-column: span 4;
  }
}

@media screen and (min-width: 900px) {
  .grid__item--half {
    grid-column: span 6;
```

Option 1: vanilla CSS (or SCSS)

```
.custom-thingy {
  grid-column: 1 / -1;
  font-size: var(--step-1);
}

@media screen and (min-width: 640px) {
  .custom-thingy {
    grid-column: 1 / 6;
    padding-top: 2em;
    padding-bottom: 1em;
  }
}

@media screen and (min-width: 900px) {
  .custom-thingy {
    grid-column: 1 / 7;
```

Option 2: Container and Item components

```
src/
└── components/
    ├── Col/
    │   ├── Col.module.css
    │   └── Col.tsx
    └── Grid/
        ├── Grid.module.css
        └── Grid.tsx
```

Grid.tsx

```
import { ReactNode, createElement } from "react";
import styles from "./Grid.module.scss";

interface GridProps extends React.HTMLProps<htmlelemer
  className?: string;
  children: ReactNode;
  tag?: keyof JSX.IntrinsicElements;
}

export default function Grid({
  className = "",
  children,
  tag = "div",
  ...props
}: GridProps) {
  const Wrapper = tag;
```

Col.tsx

```
import { ReactNode, createElement } from "react";
import cn from "classnames";
import styles from "./Col.module.scss";

interface ColProps extends React.HTMLProps<htmlelement> {
  className?: string;
  children: ReactNode;
  colWidth?: "full" | "half" | "third" | "quarter";
  tag?: keyof JSX.IntrinsicElements;
}

export default function Col({
  className = "",
  children,
  colWidth,
  tag = "div",
```

Col.module.css

```
.full,
.half,
.third,
.quarter {
  grid-column: 1 / -1;
}

@media screen and (min-width: 640px) {
  .quarter {
    grid-column: span 4;
  }
}

@media screen and (min-width: 900px) {
  .half {
    grid-column: span 6;
```

CustomThingy.module.scss

```scss
p.customThingy {
  grid-column: 1 / -1;
  font-size: var( --step-1);
}

@media screen and (min-width: 640px) {
  p.customThingy {
    grid-column: 1 / 6;
    padding-top: 2em;
    padding-bottom: 1em;
  }
}

@media screen and (min-width: 900px) {
  p.customThingy {
    grid-column: 1 / 7;
```

Option 3: Using Tailwind classes

⚠️ Yet Another Disclaimer ⚠️

The following opinion may or may not oppose your view on the matter, and that is PERFECTLY FINE. You are absolutely free to agree, disagree or not care at all.

tailwind.config.js

```
module.exports = {
  theme: {
    screens: {
      xs: "320px",
      sm: "640px",
      md: "900px",
      lg: "1200px",
      xl: "1600px",
      maxSm: { max: "639px" },
      maxMd: { max: "899px" },
      btwSmMd: { min: "640px", max: "899px" }
    },
  },
  prefix: "tw-"
};
```

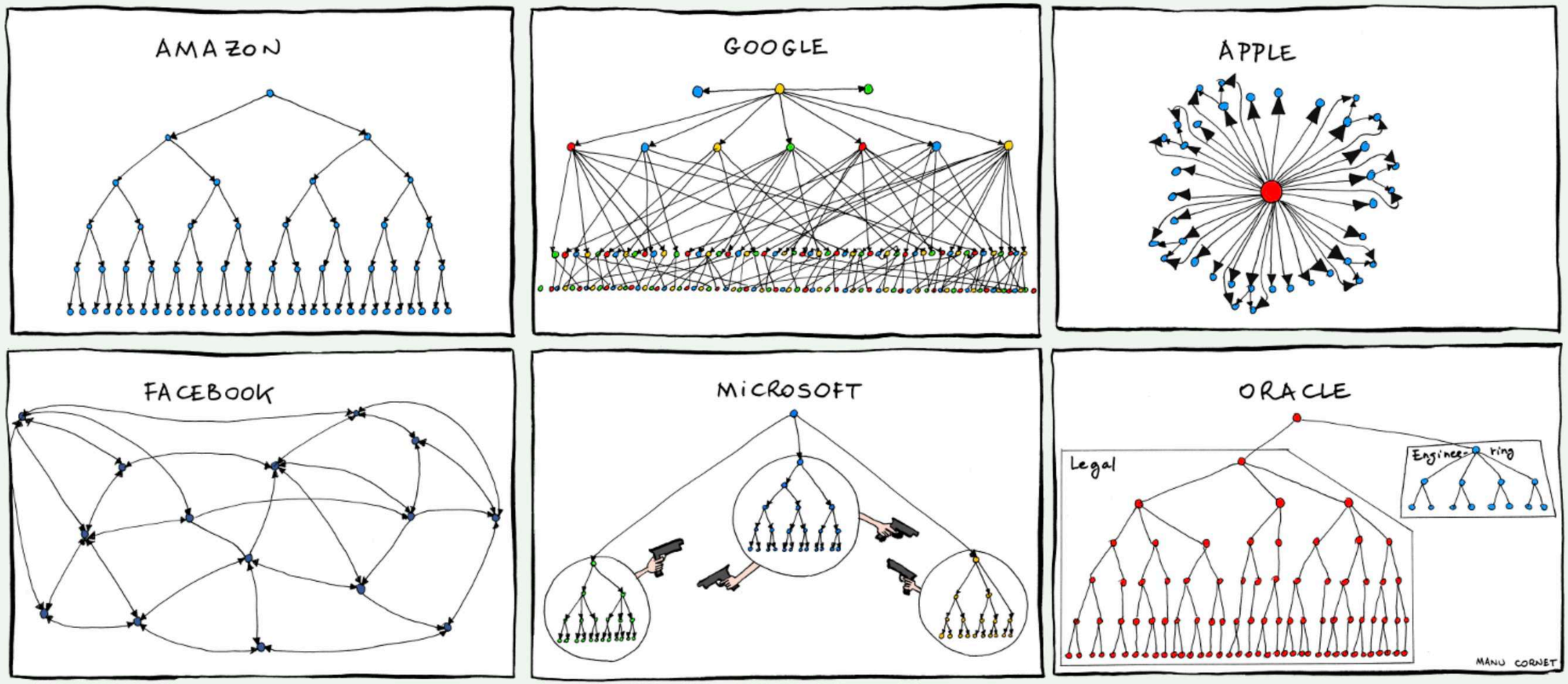
TailwindThingy.tsx

```
export default function TailwindThingy {
  return (
    <section classname="tw-grid xs:tw-grid-cols-4 sm:tw-grid-cols-8 md:tw-grid-cols-12 xs:tw-gap-3 lg:tw-gap-4
xs:tw-mx-3 sm:tw-mx-[30px] md:tw-mx-[50px] lg:tw-mx-[90px] xl:tw-mx-[180px]">
      <p classname="tw-col-span-full">Option 3: Use Tailwind classes</p>
      <p classname="tw-col-span-full md:tw-col-span-6">Well, this is spicy</p>
      <p classname="tw-col-span-full md:tw-col-span-6">
        FWIW, Tailwind has managed to support grid fairly well in this latest
        version
      </p>
      <p classname="tw-col-span-full md:tw-col-span-4">
        You will have to learn the tailwind classes to use them correctly
      </p>
      <p classname="tw-col-span-full md:tw-col-span-4">
        This basic example is able to match the previous 2 options
      </p>
```

But does it work?

CodeSandbox demo

1.00



Source: Manu Cornet, 2011-06-27 edition of Bonkers World (modified to fit slide)

So you want to introduce Grid to your application?

- Are there **preferred technologies** used within the organisation?
- How is **big** is your application and how is it **structured**?
- How **flexible** does the design system need to be?
- Are there cases where code is contributed by **new developers** often?
- What is the **documentation** culture like in your organisation?

So you want to introduce Grid to your application?

- Who is responsible for the **maintenance** and development of **new components or pages** on the application?
  - Is it a small team of full-time developers overseeing the entire project?
  - Is it numerous teams responsible for their own respective set of components and pages?
  - What is the overall CSS skill level of the developers contributing to the codebase?
  - Are the contributing developers very familiar with the frameworks and libraries used in the codebase?

Document the "Why"

> *One size does not fit all*
>
> *–Frank Zappa*

감사합니다

https://chenhuijing.com

@hj_chen

@huijing

@huijing

Font is CookieRun by Devsisters.