WOULD YOU SAY THAT CSS IS AWESOME?

Joonas Kallunki

Visuel Developer, Frontend Stapler

USER PATIENCE

Time for a page to be interactive is smaller with UI styled by CSS than JS.

> **...the browser won't render any processed content until the CSSOM is constructed.**

Google Web Fundamentals

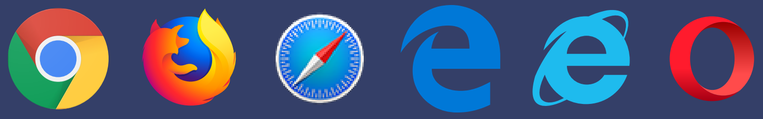
> **Performance is about culture supported by tools. Browsers optimize HTML+CSS as much as possible. Moving more of your work into JS puts the burden on your team and their tools**

Alex Russell
(The Cost Of JavaScript In 2018 - Addy Osmani)

> **The** transform **and** opacity **properties are guaranteed to neither affect nor be affected by the normal flow or DOM environment (that is, they won’t cause a reflow or repaint, so their animation can be completely offloaded to the GPU).**

Sergey Chikuyonok,
CSS GPU Animation: Doing It Right,
Smashing Magazine

CSS GRID



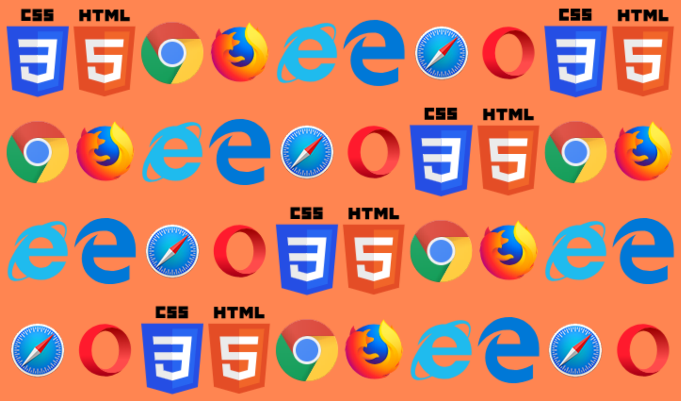
57 52 10.1 16 11* 44

All mobiles have now support *(even Samsung Internet)* *~~(Opera Mini)~~*

** **grid-template-areas** is your new best friend (autoprefixer)*

CSS GRID

- It is responsive
- Use it together with flexbox and other goodies



```
<div class="grid">
  <div class="item"><img src="CSS3_logo_and_wordmark.svg" /></div>
  <div class="item"><img src="HTML5_logo_and_wordmark.svg" /></div>
  <div class="item"><img src="Google_Chrome_icon_%28September_2014%29.svg" /></div>
  <div class="item"><img src="Firefox_Logo%2C_2017.svg" /></div>
  <div class="item"><img src="Internet_Explorer_10%2B11_logo.svg" /></div>
  ...
</div>
```

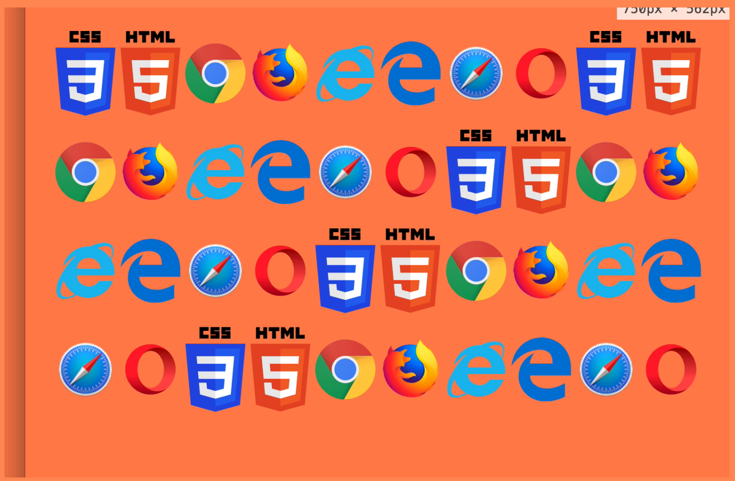
```
.grid {
  display: grid;
  grid-template-columns: repeat(auto-fit, minmax(60px, 1fr));
```

- Repeat unspecified number of columns
- Fit the currently available columns into the space
- Minimum of 60 pixels and a maximum of 1 fraction unit of the available space

```
  grid-auto-rows: 100px;
}
```

- Set implicitly created rows height to 100 pixels

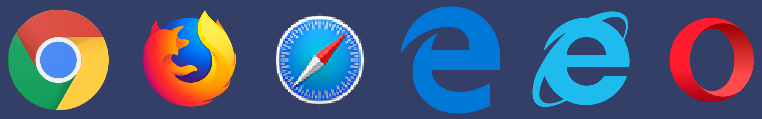
BOOM!



```
.grid {
  display: grid;
  grid-template-columns: repeat(auto-fit, minmax(60px, 1fr));
  grid-auto-rows: 100px;
}

<div class="grid">
  <div class="item"><img src="CSS3_logo_and_wordmark.svg" /></div>
  <div class="item"><img src="HTML5_logo_and_wordmark.svg" /></div>
  <div class="item"><img src="Google_Chrome_icon_%28September_2014%29.svg" /></div>
  <div class="item"><img src="Firefox_Logo%2C_2017.svg" /></div>
  <div class="item"><img src="Internet_Explorer_10%2B11_logo.svg" /></div>
  ...
</div>
```

CSS CUSTOM PROPERTIES

Also known as CSS variables

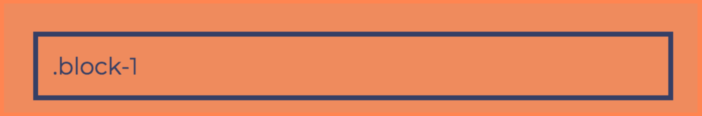


49 31 9.1 16 **NO** 36

CSS CUSTOM PROPERTIES

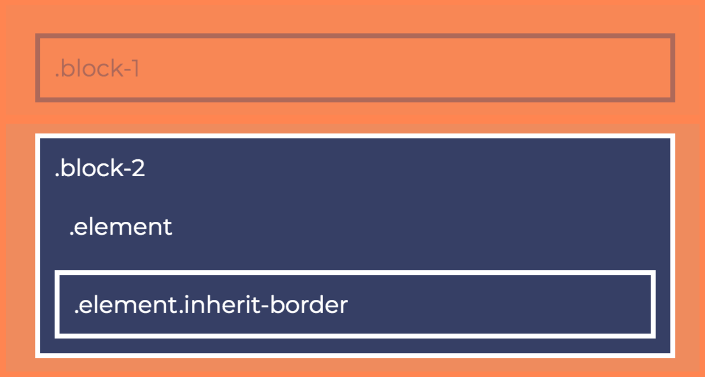
- Declarative - no imperative order of operations
- Can be redefined with inline styles dynamically

(For example implement mouse coordinates, scroll position or user input with JS)



```
.block-1 {
  border-color: currentColor;
  color: #343f68;
}
```

- Supported starting from IE9



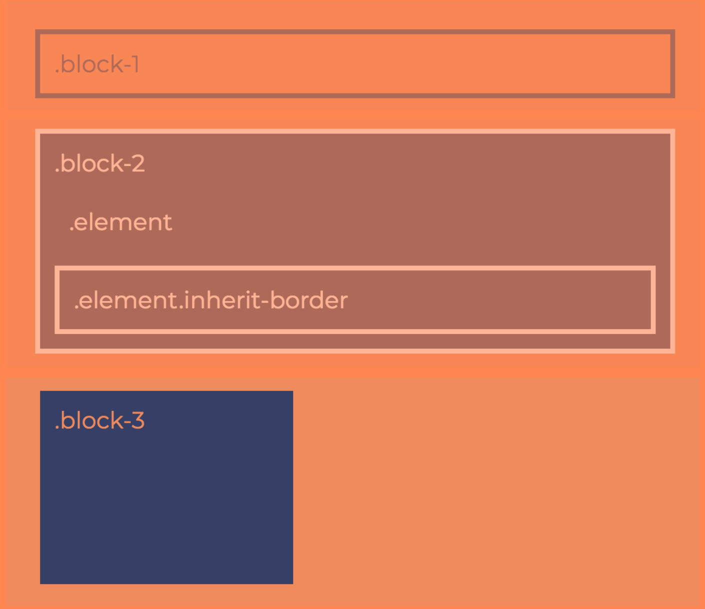
```
.block-1 {
  border-color: currentColor;
  color: #343f68;
}
```

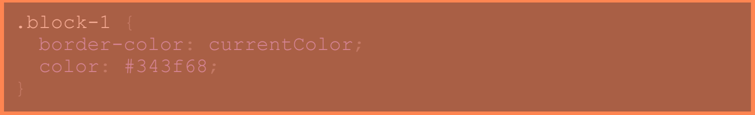
```
.block-2,
.element {
  border: var(--border);
  --border: initial;
```

- Set to inital instead of the default inherit

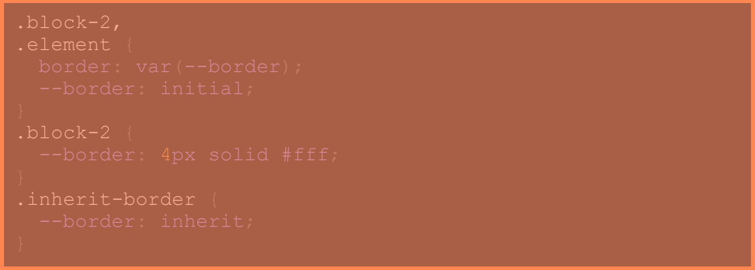
```
}
.block-2 {
  --border: 4px solid #fff;
}

.inherit-border {
  --border: inherit;
}
```

- Inherit if needed

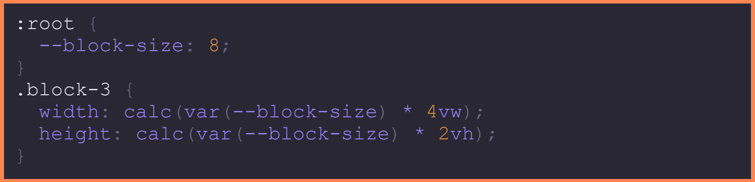




```
.block-1 {
  border-color: currentColor;
  color: #343f68;
}
```

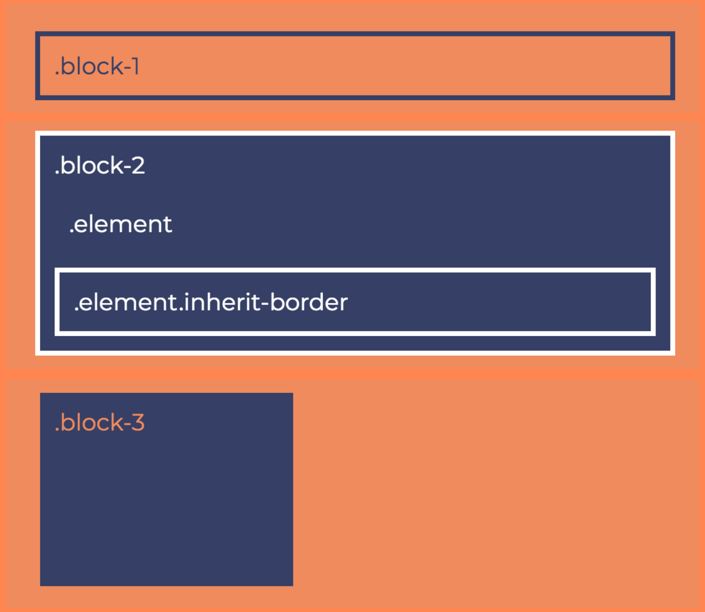


```
.block-2,
.element {
  border: var(--border);
  --border: initial;
}
.block-2 {
  --border: 4px solid #fff;
}
.inherit-border {
  --border: inherit;
}
```



```
:root {
  --block-size: 8;
}
.block-3 {
  width: calc(var(--block-size) * 4vw);
  height: calc(var(--block-size) * 2vh);
}
```

- Use Calc()-function for calculations
- In HTML, :root represents the <html> element, except that its specificity is higher.

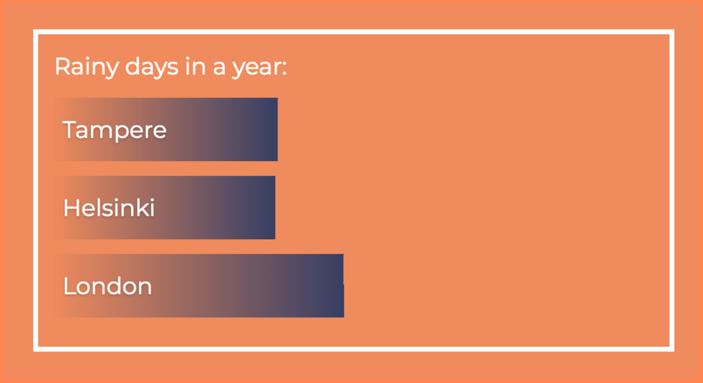


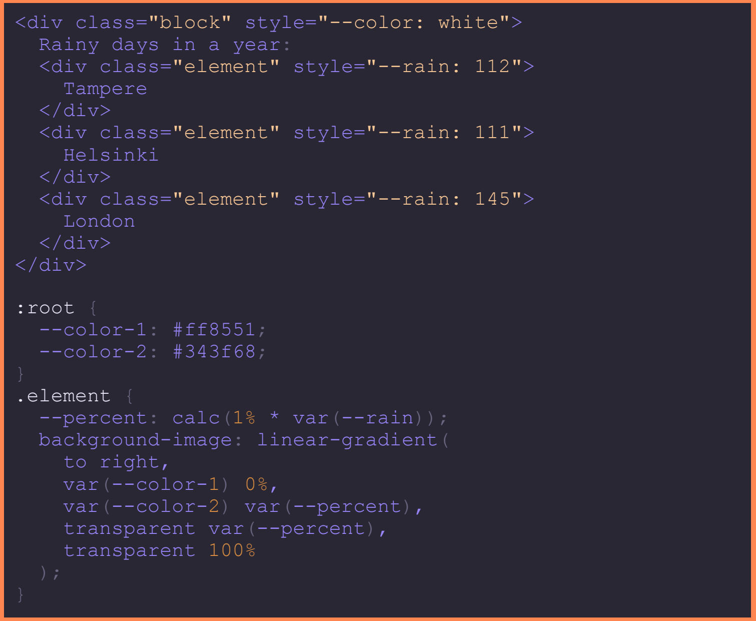
```css
.block-1 {
  border-color: currentColor;
  color: #343f68;
}
```

```css
.block-2,
.element {
  border: var(--border);
  --border: initial;
}
.block-2 {
  --border: 4px solid #fff;
}
.inherit-border {
  --border: inherit;
}
```

```css
:root {
  --block-size: 8;
}
.block-3 {
  width: calc(var(--block-size) * 4vw);
  height: calc(var(--block-size) * 2vh);
}
```

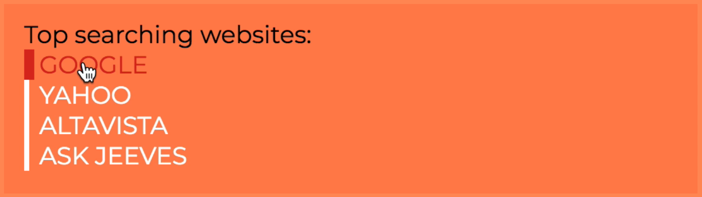
- Use Calc()-function for calculations
- In HTML, :root represents the <html> element, except that its specificity is higher.





```
<div class="block" style="--color: white">
  Rainy days in a year:
  <div class="element" style="--rain: 112">
    Tampere
  </div>
  <div class="element" style="--rain: 111">
    Helsinki
  </div>
  <div class="element" style="--rain: 145">
    London
  </div>
</div>

:root {
  --color-1: #ff8551;
  --color-2: #343f68;
}
.element {
  --percent: calc(1% * var(--rain));
  background-image: linear-gradient(
    to right,
    var(--color-1) 0%,
    var(--color-2) var(--percent),
    transparent var(--percent),
    transparent 100%
  );
}
```

- `style.setProperty('--var', value)`



Top searching websites:

GOOGLE

YAHOO

ALTAVISTA

ASK JEEVES

```
.link:hover:before {
  border-left: 8px solid gray;
}
```

- Degrade gracefully

```
@supports (--css: variables) {
  .link:hover {
    color: var(--hover-color, #343f68);
  }
  .link:hover:before {
    border-left-color: var(--hover-color, gray);
  }
```

- Second argument to var() is the default value

```
  .google {
    --hover-color: #d62d20;
  }
  .yahoo {
    --hover-color: #410093;
  }
}
```

- Just degrade gracefully