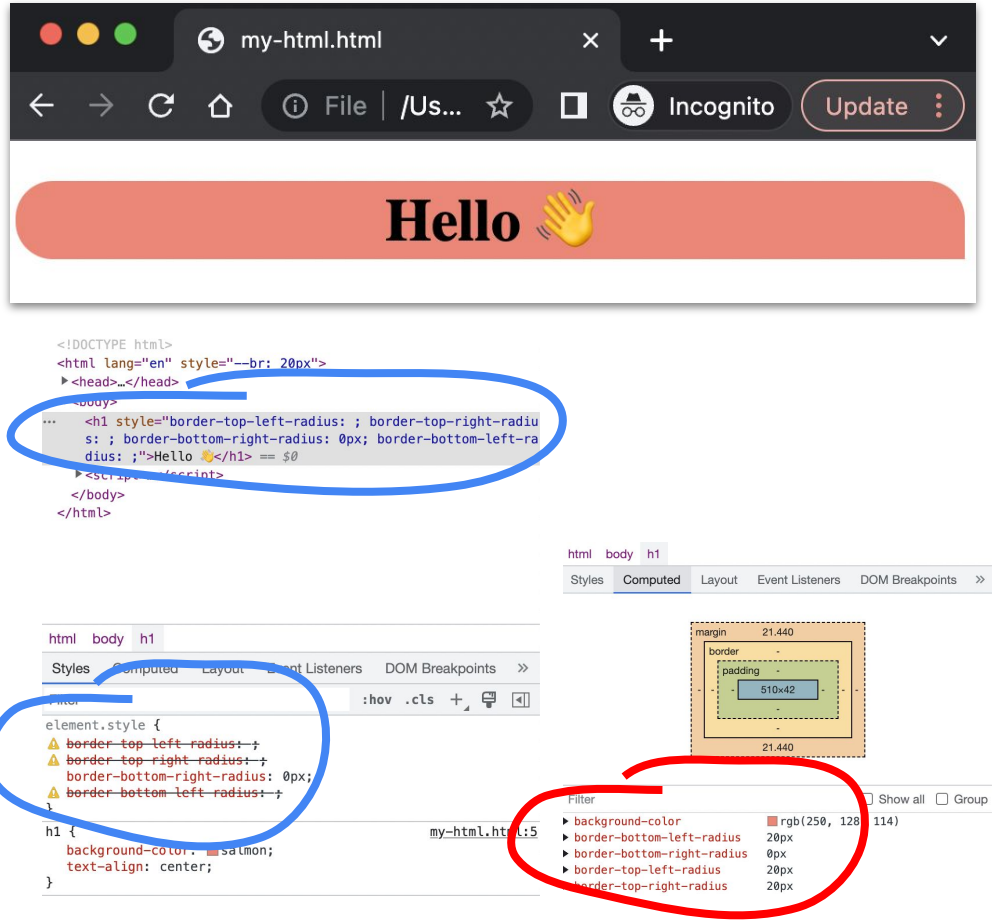
CSS Variables in theming support

and the consequences of thereof

Demo time

```
<!DOCTYPE html>
<html lang="en" style="--br: 20px">
  <head>
    <style>
      h1 {
        background-color: salmon;
        text-align: center;
      }
    </style>
  </head>
  <body>
    <h1>Hello 👋</h1>
    <script>
      const h1 = document.querySelector("h1");

      window.addEventListener("load", () => {
        h1.style.borderRadius = "var(--br)";
        h1.style.borderBottomRightRadius = 0;
      });
    </script>
  </body>
</html>
```



There are three main stages(*) in resolving a css value

(*there are actually more but these are the main ones)

Parsed time (specified values)
When the actual CSS code is read, parsed, and converted to a tree of objects (CSSOM). One and done operation.
(This is when duplicate properties are removed, and the shorthand properties are expanded)

There are three main stages(*) in resolving a css value

(*there are actually more but these are the main ones)

Computed values

calculated from the specified value by:
- handling the special values inherit, initial, revert, revert-layer, and unset.
- converting relative values (such as those in em units) to absolute values.

(This is when relative - e.g. viewport units are resolved to px)

There are three main stages(*) in resolving a css value

(*there are actually more but these are the main ones)

Used values

The final stage, when values are fully resolved to be used for painting.
(This is % widths are resolved to px)

Handling invalid custom properties

When the values of custom properties are parsed, the browser doesn't yet know where they will be used, so it must consider nearly all values as *valid*.

Unfortunately, these valid values can be used, via the var() functional notation, in a context where they might not make sense. Properties and custom variables can lead to invalid CSS statements, leading to the new concept of *valid at computed time*.

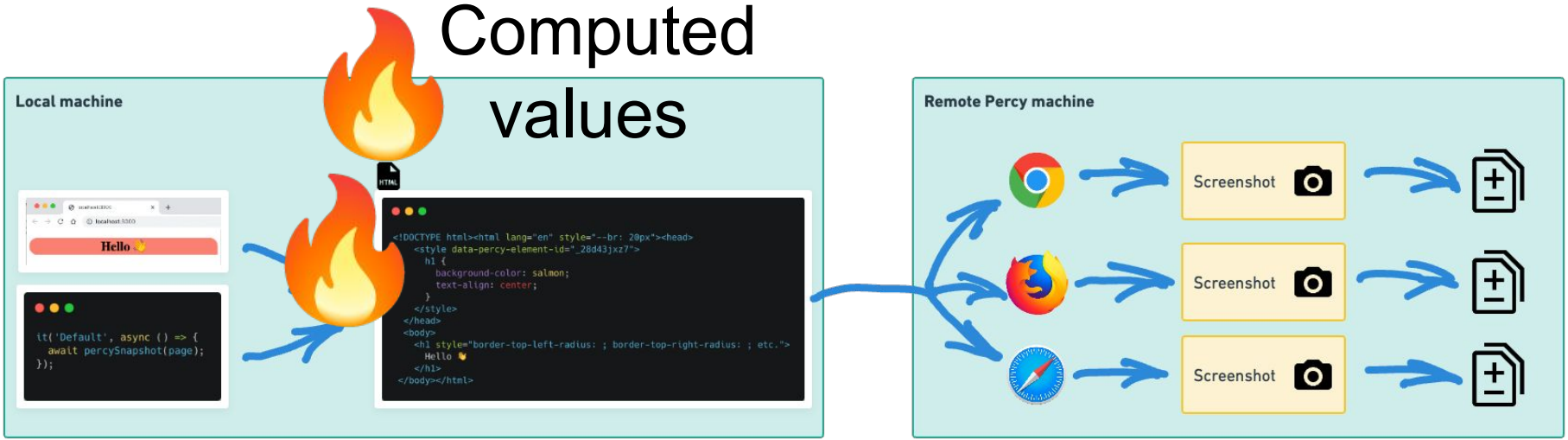
MDN

Handling invalid custom properties

```
--primary-color: 16px;
background: var(--primary-color) unset
```

Anytime a property becomes invalid at computed value time, it’s essentially set to `unset`.

How Percy works?



Tech debt

```
// TODO: uncomment when issue with Percy is resolved

// await percySnapshot(page, 'ThemeProvider - after global theme change');

// await percySnapshot(page, 'ThemeProvider - after local theme change');

// await percySnapshot(page, 'AsyncCreatableSelectField - Open');

// await percySnapshot(page, 'AsyncSelectField - withDefaultOptions -
open');

// await percySnapshot(page, 'CreatableSelectInput - open');

// await percySnapshot(page, 'DateInput - open');
```

Now what?

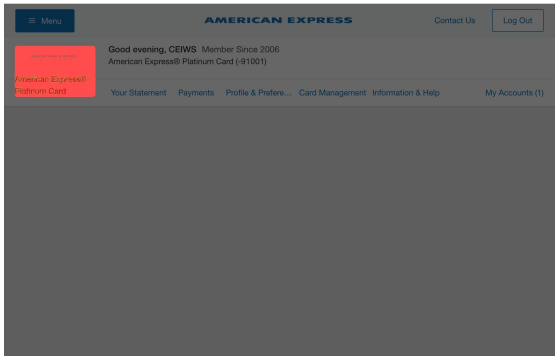
Percy replacement?

Admittedly, DX with Percy is good

Keep using Percy

❌ `border`
❌ `border-radius`

jest-image-snapshot ?



Thank you!