Responsive Type

Previously we learned about fluid units of measure. In this lecture we will look more into how responsive type works as well as expand on more methods to mitigate our typography.

Font Size on the HTML

(The default font size on the HTML is 16px)

Let’s take a look at what happens if we put a font size on the HTMl Selector:

```
<style>
html {
    font-size: 10px;
}


</style>

<body>
    <h1>H1 Text</h1>
    <h2>H2 Text</h2>
    <h3>H3 Text</h3>
    <h4>H4 Text</h4>
    <h5>H5 Text</h5>
    <h6>H6 Text</h6>

    <p>Lorem ipsum dolor sit amet consectetur adipisicing elit. Quas quae accusamus ea iste ad sit hic ex
```

Result:

H1 Text

H2 Text

H3 Text

H4 Text

H5 Text

H6 Text

Lorem ipsum dolor sit amet consectetur adipisicing elit. Quas quae accusamus ea iste ad sit hic expedita quia? Vero laboriosam amet voluptatem placeat unde praesentium doloribus. Quis cupiditate esse sit.

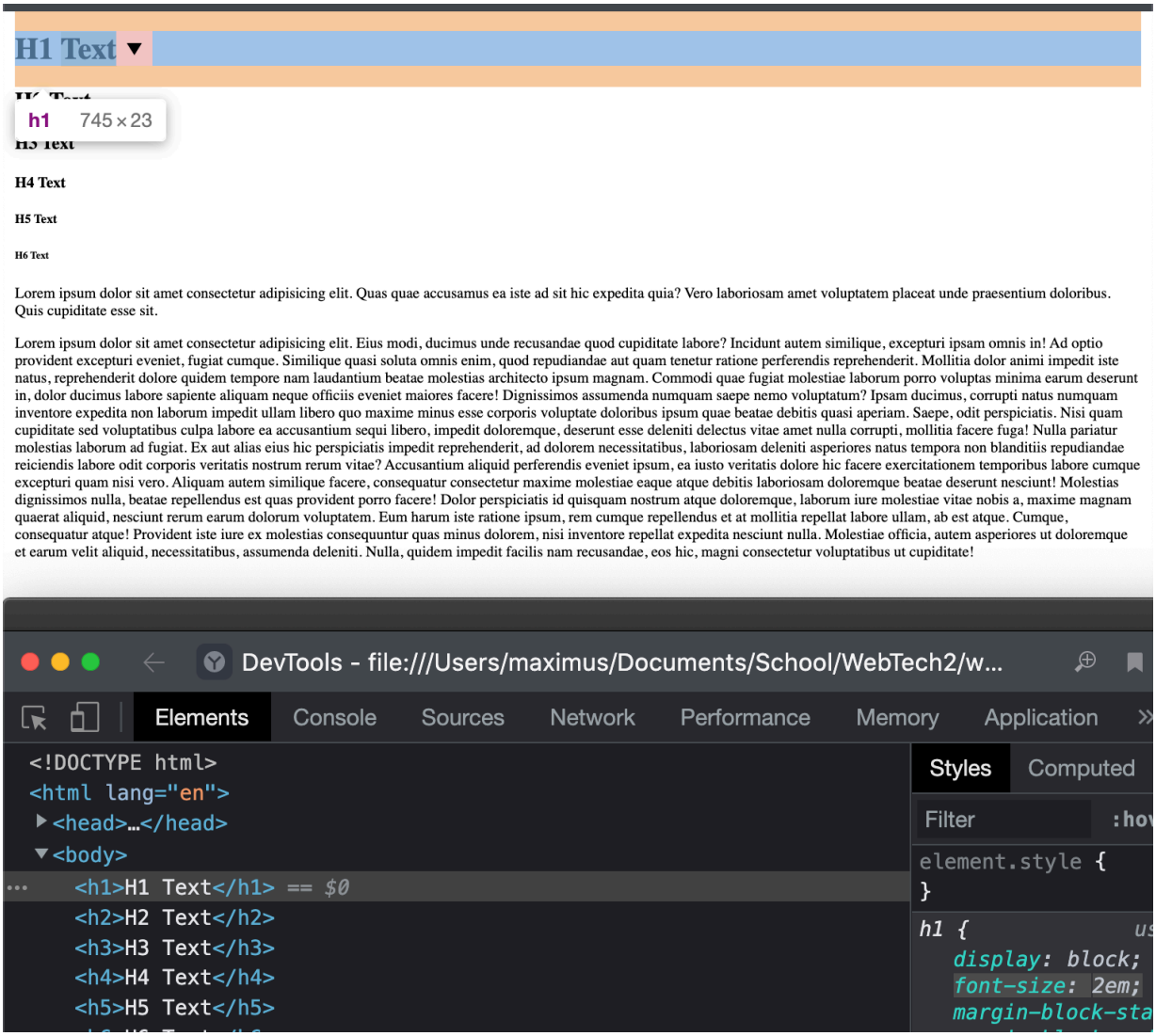
Lorem ipsum dolor sit amet consectetur adipisicing elit. Eius modi, ducimus unde recusandae quod cupiditate labore? Incidunt autem similique, excepturi ipsam omnis in! Ad optio provident excepturi eveniet, fugiat cumque. Similique quasi soluta omnis enim, quod repudiandae aut quam tenetur ratione perferendis reprehenderit. Mollitia dolor animi impedit iste natus, reprehenderit dolore quidem tempore nam laudantium beatae molestias architecto ipsum magnam. Commodi quae fugiat molestiae laborum porro voluptas minima earum deserunt in, dolor ducimus labore sapiente aliquam neque officiis eveniet maiores facere! Dignissimos assumenda numquam saepe nemo voluptatum? Ipsam ducimus, corrupti natus numquam inventore expedita non laborum impedit ullam libero quo maxime minus esse corporis voluptate doloribus ipsum quae beatae debitis quasi aperiam. Saepe, odit perspiciatis. Nisi quam cupiditate sed voluptatibus culpa labore ea accusantium sequi libero, impedit doloremque, deserunt esse deleniti delectus vitae amet nulla corrupti, mollitia facere fuga! Nulla pariatur molestias laborum ad fugiat. Ex aut alias eius hic perspiciatis impedit reprehenderit, ad dolorem necessitatibus, laboriosam deleniti asperiores natus tempora non blanditiis repudiandae reiciendis labore odit corporis veritatis nostrum rerum vitae? Accusantium aliquid perferendis eveniet ipsum, ea iusto veritatis dolore hic facere exercitationem temporibus labore cumque excepturi quam nisi vero. Aliquam autem similique facere, consequatur consectetur maxime molestiae eaque atque debitis laboriosam doloremque beatae deserunt nesciunt! Molestias dignissimos nulla, beatae repellendus est quas provident porro facere! Dolor perspiciatis id quisquam nostrum atque doloremque, laborum iure molestiae vitae nobis a, maxime magnam quaerat aliquid, nesciunt rerum earum dolorum voluptatem. Eum harum iste ratione ipsum, rem cumque repellendus et at mollitia repellat labore ullam, ab est atque. Cumque, consequatur atque! Provident iste iure ex molestias consequuntur quas minus dolorem, nisi inventore repellat expedita nesciunt nulla. Molestiae officia, autem asperiores ut doloremque et earum velit aliquid, necessitatibus, assumenda deleniti. Nulla, quidem impedit facilis nam recusandae, eos hic, magni consectetur voluptatibus ut cupiditate!

As we can see, the text is all smaller, even the h1 to h6 text. Why? Because by default they are sized by em units.

h1 = 2em
H2 = 1.5em
H3 = 1.17em
H4 = 1.33em
H5 = 0.83em
H6 = 0.67em
P = 1em

So everything will resize itself

based on what we tell the HTML font size in this case.



We *could* use a VW for the font size. But if we did there would be a trade off. Although it would be responsive it will also destroy the ability to zoom in and out using CMD+ or CMD– .

```
<style>
html {
    font-size: 3vw;
}


</style>

<body>
    <h1>H1 Text</h1>
    <h2>H2 Text</h2>
    <h3>H3 Text</h3>
    <h4>H4 Text</h4>
    <h5>H5 Text</h5>
    <h6>H6 Text</h6>

    <p>Lorem ipsum dolor sit amet consectetur adipisic
```

Calc

Instead we can use the CALC way. In CSS you can when needed, use a calculation to do things.

```
<style>

html {
    font-size: calc(.5vw + 1em);
}

</style>

<body>
    <h1>H1 Text</h1>
    <h2>H2 Text</h2>
    <h3>H3 Text</h3>
    <h4>H4 Text</h4>
    <h5>H5 Text</h5>
    <h6>H6 Text</h6>

    <p>Lorem ipsum dolor sit amet consectetur adipisicing elit.
```

In this case, the font size is calculated to be .5 + 1em. Which is .5% of the viewport width + 1em (which is 16px by default).

Using this, we will have fluid text that will resize as the window resizes.

We could still style different aspects of the font as we see fit if we needed to:

```
<style>

html {
    font-size: calc(.5vw + 1em);
}

p {
    font-size: 2em;
}

</style>

<body>
    <h1>H1 Text</h1>
    <h2>H2 Text</h2>
    <h3>H3 Text</h3>
    <h4>H4 Text</h4>
    <h5>H5 Text</h5>
    <h6>H6 Text</h6>

    <p>Lorem ipsum dolor sit amet consectetur adipisicing elit.
```

This would still result in a fluid text that would resize as the window resizes.

Clamp

Clamp is a new method we can use to deal with sizing. What it does is it “clamps” a size between a min and a max size. The middle number is the

preferred value. Here is an example:

```
.box {
    height: 100px;
    width: clamp(200px, 20%, 400px);
    background-color: red;
}
```

Here we can say a minimum size and a maximum size in between a preferred size. In our example, we have a box that will stay in between 200px and 400px but will be fluid in between those 2 values.

We can also use this for font sizes:

```
p {
    font-size: clamp(1em, 1vw, 3em);
}
```

In this case, the font size will never be smaller then 1em and never bigger than 3em but it will resize to

stay 1vw (1% of the viewport width).

This is much easier than using the calc function.

Scalable Class Names

Another approach to scalability is reusing classnames. In the above section, we have the case where we have a border, Which can be re-deployed when needed. We could do the same thing via classname instead.

As easy as variables are to deploy, so too are class names. Just put the reusable thing on its own selector in the CSS.

```
.border {
    border: 2px solid #222;
}
```

Then you can deploy the class anywhere you need it in the HTML.

```
<div class="box border">

</div>
```

All this is doing is using the fact that we can share classes and give 1 element multiple classes to our advantage.

Variables

Using CSS we can create variables. These allow us to easily scale certain repetitive aspects of our CSS. Need a certain color for a brand distributed through your site? How about a reoccurring border? One way to easily deploy this is to use a CSS variable.

One thing to note, variables have to be declared in a selector, we can make variables in a `:root { }` selector. The reason why we might

want to do this, is because any variable in the :root { } can be used anywhere on the entire site because it is usable anywhere in the entire HTML.

If we were to make a variable in the .header{} selector for example, it would only be usable in that part of the page.

Here is an example:

```
:root {
    --border: 2px solid #222;
    --maincol: rgb(11, 120, 230);
}
```

--name: value;

To use a variable, we declare the variable using a --varname: After which we can write the value.

After the variable is declared, we can deploy it by writing the property

and then giving it the var value and calling the aforementioned variable name.

It’s that easy to deploy variables.

```
.box {
    height: 100px;
    width: clamp(200px, 20%, 400px);
    background-color: var(--maincol);
    border: var(--border);
}
```