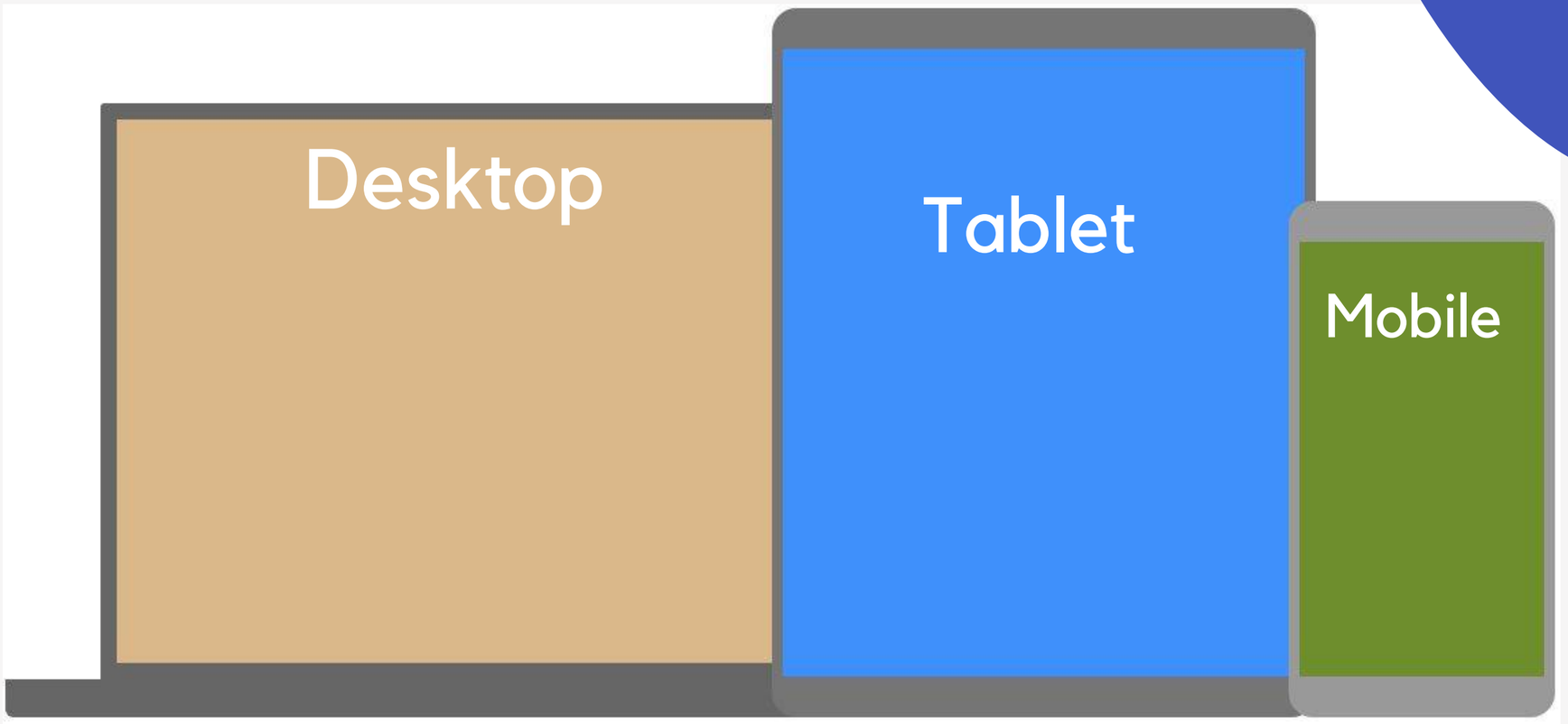
CSS RESETS & MEDIA QUERIES



CSS Resets

What is it and why are they used?

Since we strive to develop the same functionality/style across all browsers, it's nice to start with a level playing field for all browsers. Unfortunately, different browsers make slightly different assumptions about the default styling of a webpage. For example, in Chrome, the body has a default margin of 8px on all sides.

In order to eliminate possible discrepancies across browsers, it's very common to use what's called a CSS Reset. This is just a CSS stylesheet that attempts to overwrite (or reset) any default browser styles with styling that will then be the same across different environments.

Common CSS Resets

You typically don't write a CSS Reset yourself; instead, there are a few commonly used resets you can choose from. Here are a couple:

1. reset.css - this was created by Eric Meyer and it basically resets/clears everything. It's a bit aggressive, but if you want to start from scratch it's a great tool. http://meyerweb.com/eric/tools/css/reset/
2. normalize.css - https://necolas.github.io/normalize.css/

normalize.css is quite common especially when dealing with responsive design. In order to maintain the same styling and display across all browsers, it's essential to make sure that a standard is set.

Reset vs Normalize

- CSS resets aim to **remove** all built-in browser styling. Standard elements like h1 - h6, p, strong, em end up looking exactly alike, having no decoration at all. You're then supposed to add all decoration yourself.

- Normalize.css aims to make built-in browser styling **consistent** across browsers. Elements like h1 - h6 will appear bold, larger et cetera in a consistent way across browsers. You're then supposed to add only the difference in decoration your design needs.

Is it necessary?

Typically, you would choose one or the other.

Normalize explicitly sets everything to have some general spacing and predictable behaviors, like sensible font sizes and margins across the board. The reset version takes everything down to no spacing (margin or padding) and everything down to 16px font size; sometimes it could be considered a little overzealous.

If a site is primarily layout-driven, Reset works well, because you'd have to remove margins off of everything otherwise. Whereas if the site is primarily content-driven (like a blog or news site), Normalize might be a better choice since it would make most content combinations have a minimum level of spacing consistency—especially for things you might not think to style: like the spacing above a paragraph that comes after an h6 inside of a doubly nested list inside of blockquote...

In Short,

If you want all the styles, including margin and padding reset across all browsers, use **reset. css**. Then apply all decorations and stylings yourself. If you simply like the built-in stylings but want more cross-browser synchronicity i.e. normalizations then use **normalize.**

- A reset removes all decoration leaving it up to the developer to add. This can mean more work and more css.
- That can be good or bad depending on your requirements.A normalize basically just makes things consistant. The developer then only needs to add css to implement a design.

However, we don't necessarily need a heavy-handed reset, or even a reset at all, because CSS browser compatibility issues are much less likely than they were in the old IE 6 days.

Are there other options?

Once you know what the reset does, you can honestly get away with just taking care of the base concerns (removing spacing) with something as simple as this at the top of your stylesheet:

```
*,
*::before,
*::after {
    margin:0;
    padding:0;
    box-sizing:border-box;
}
```

```
*,*::after,
*::before {
margin: 0;
padding: 0;
box-sizing: inherit;
}
html {
font-size: 62.5%;
}
body {
box-sizing: border-box;}
```

Media Queries

What are Media Queries?

Media Queries are a feature in CSS3 that allow you to specify when certain CSS rules should be applied. This allows you to apply different CSS styles for different environments such as viewport size and device type:

- Mobile devices
- Tablets & large smartphones
- Laptops, larger tablets, and small desktops
- Large Desktops and monitors.

Media Queries

Anatomy of a Media Query

The simplest media query syntax looks like this:

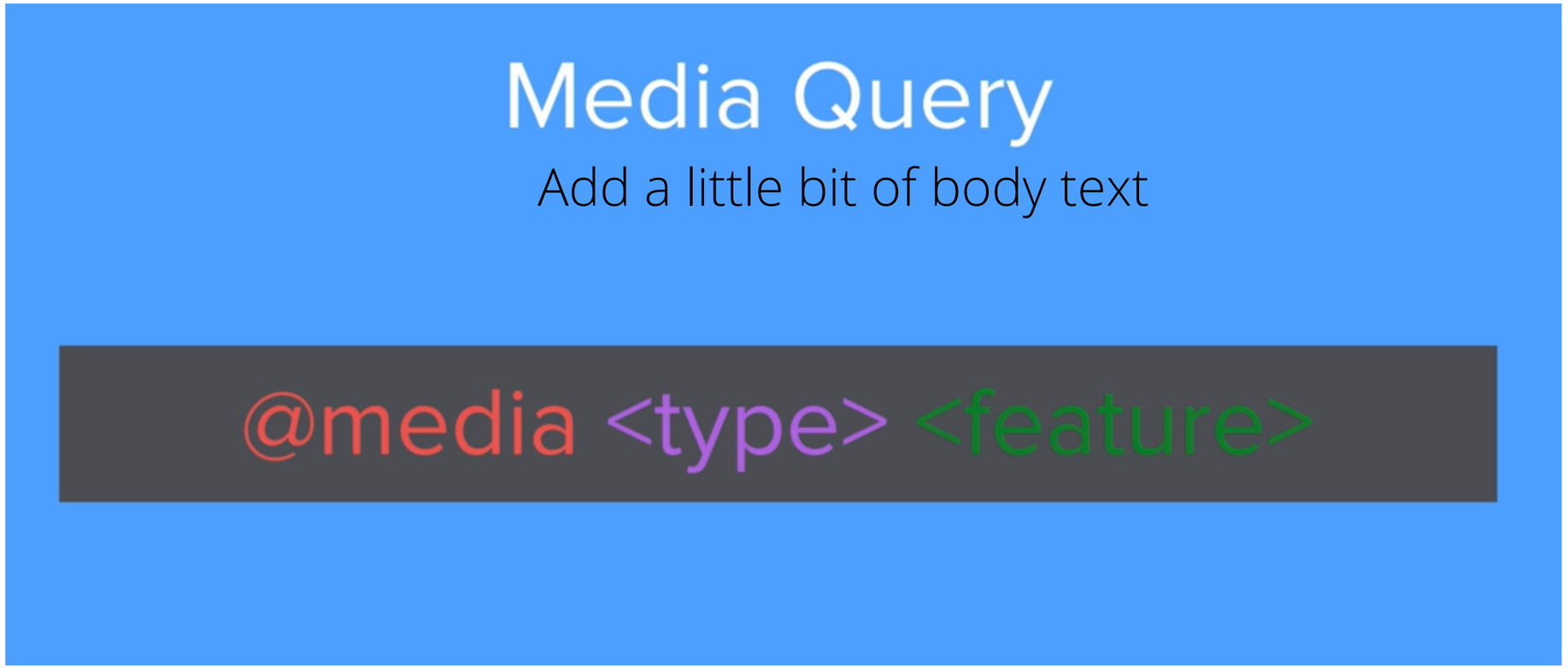
```
@media media-type and (media-feature-rule) {
  /* CSS rules go here */
}
```

It consists of:

- A media type, which tells the browser what kind of media this code is for (e.g. print, or screen).
- A media expression, which is a rule, or test that must be passed for the contained CSS to be applied.
- A set of CSS rules that will be applied if the test passes and the media type is correct.

Media Queries

Anatomy of a Media Query



Media Queries

Anatomy of a Media Query

```
@media screen (min-width: 900px) {

//Change Something

}
```

Media Queries

Anatomy of a Media Query

```
@media screen (min-width: 900px) {
                (max-width: 1200px)

//Change Something


}
```

Media Queries

Media Types

The types of Media you can specify are:

- all (default)
- print
- screen
- speech

Media Queries

Targeting Device Orientation

```
@media screen and ( ... ) {

}

@media print and ( ... ) {

}
```

Style for Device Screens

Style for when a user is printing the page

Media Queries

Targeting Device Orientation

```
@media (orientation: landscape) {

}

@media (orientation: portrait) {

}
```

the width is greater than the height.

the height is greater than or equal to the width.

Media Queries

What's a Breakpoint

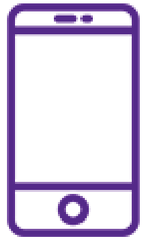
In responsive design, the widths that we target in our media queries are called breakpoints, because that is the point at which we will change the CSS rules.

Don't worry about creating tons of media queries with breakpoints targeting every device width that exists.This was a practice back in the early days of responsive design.

But nowadays, with so many different phones, tablets, and screen sizes, it’s simply not practical. You’ll only end up with a confusing (and inefficient) number of media queries.

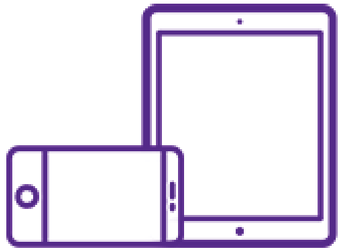
Media Queries

Common Breakpoints



0-480

Smaller smartphones

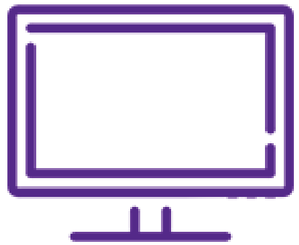


481-768

Tablets & larger smartphones

769-1279

Laptops, larger tablets in landscape, and small desktops



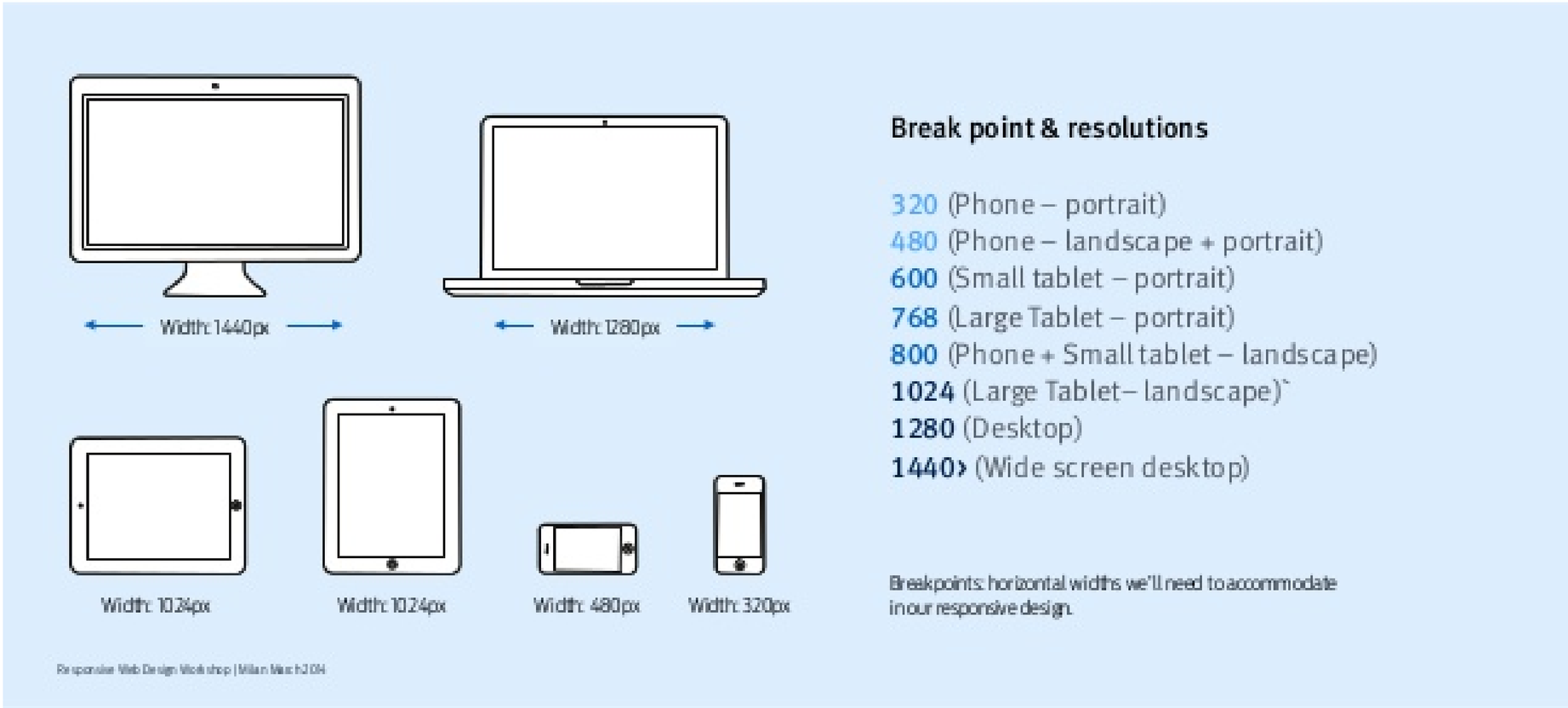
1280+

Larger desktops and monitors

Media Queries

Common Breakpoints

Breakpoints & Media queries



Media Queries

Mobile Vs Desktop

```
@media (min-width: 600px) {
    */ mobile first /*
}

@media (max-width: 600px) {
    */ desktop first /*
}
```

Targeting Smaller Screens First

Targeting Bigger Screens First

Media Queries

Should you use min-width or max-width?

Using min-width would be considered the mobile first method.
This way, you'd be setting the mobile styles as your default styles.

You can of course choose whichever makes the most sense to you and it depends on your users as well. These days, the majority of users are on a mobile device.

Media Queries

Which unit should you use?

There's a lot of debate on this but the general consensus is that EM's perform more consistently across browsers.

This will of course depend on the users and if you care about supporting older browsers.

VW seems nice but it will override the users zoom capabilities.

Generally speaking, It’s hard to obtain an intuition on the way zoom levels and user preferences interact with one another.

Media Queries

Mobile Vs Desktop

// Small devices (landscape phones, 576px and up)

@media (min-width: 576px) { ... }

// Medium devices (tablets, 768px and up)

@media (min-width: 768px) { ... }

// Large devices (desktops, 992px and up)

@media (min-width: 992px) { ... }

// Extra large devices (large desktops, 1200px and up)

@media (min-width: 1200px) { ... }

Media Queries

Mobile Vs Desktop

@media (max-width: 575.98px) { ... }

// Small devices (landscape phones, less than 768px)

@media (max-width: 767.98px) { ... }

// Medium devices (tablets, less than 992px)

@media (max-width: 991.98px) { ... }

// Large devices (desktops, less than 1200px)

@media (max-width: 1199.98px) { ... }