Information Technology 3

Typesetting for the Web

Made with Agnes.js

Kristof Michiels

Typesetting for the web is an art

The importance of typesetting

- As a reader, you know the importance of well formatted, expertly styled text
- Good typesetting makes the act of reading recede into the background. You're not scanning letters on a page, you're visualizing worlds
- Skillful typesetting allows readers to absorb information quickly
- Good typesetting is often invisible
- As a web designer, you should be sensitive to how text is used in design examples

Typesetting for the Web

- Typesetting for the Web is really important
- Important concept: typography driven design
- You start your design from text
- It might be looked at as the secret cornerstone of flexible, responsive design
- This week we'll make a web page wherein typography plays an important role

The nature of the text

- The nature of the text you will be using is really important
- Take responsibility for the text
- You have to care about the text in order to make a good web design out of it
- With real-world projects: getting good text copy will sometimes be a problem
- As a general rule, if you don’t have enough text, ask for more
- If it’s not the final version, ask for an update
- If the text is going to change all the time, like news, find some representative example text
- Once you "KNOW" your text, do contextual, visual research. Understand how texts of this nature are placed in a web context. Once you know these rules, apply or break them

Clean text and declutter

- When you have selected your text, first clean it up!
- Remove tabs, indents, and any other invisible characters that do not belong in the text
- Eliminate any line breaks, hyphens in words, ... within running text, which are often the result of copy-and-paste problems

Marking up your texts

Mark the text up

- Finally, mark up the elements of the text—paragraphs, headings, lists, block quotes, and so on—as well as any strings that need emphasis
- Here your HTML-skills come into play
- Don't forget that markup is the most basic and possibly the most important typographic decision we make
- Think of it as invisible composition: markup
- It’s our core means of establishing hierarchy, patterns, and meaning

Reset default styles

- Most browsers and user agents ship with default styles
- To counter this, use reset stylesheet
- Eric Meyer has advocated the use of reset stylesheets since 2007
- We are going to use his latest version as a starting point for our exercise: https://meyerweb.com/eric/tools/css/reset/
- A reset stylesheet puts the focus on HTML elements rather than on browser behavior

Framing the composition

- The problem is that we need to frame our compositions differently for the web than for traditional media
- We don't have control over the myriad of screens our text will be read on
- We do however have one reliable building block for typesetting today: default font size

```
<meta name="viewport" content="width=device-width">
```

```
@viewport {
    width: device-width;}

:root {
    font-size: 100%;}
```

Framing the composition

- First, we need to make sure readers see our composition at its actual size on any device, without any artificial scaling, while maintaining their ability to zoom in if they want to
- We do this with CSS Device Adaptation—the @viewport rule. It is telling the browser to fit the width of our layout to the width of a person's device. This makes our text actual-size, instead of scaling it to match a specific pixel-based width
- Note that it's important to specify both the @viewport descriptor in CSS and the HTML meta tag, because browsers currently use either one or the other
- You can set the font-size to 100%, or for example to 62.5%, which makes an em or rem 10px in size

```
:root {
    font-size: 62.5%; /* makes a rem = 10px */}
```

Webfonts

Getting to webfonts

- Webfonts aren’t hard to find
- Places like Adobe Typekit or Fontstand are better than Google Fonts, but cost money. For professional designers it is not an issue
- Repositories of free and open-source fonts—even something as convenient as Google Fonts—have significantly fewer choices for text use
- In many of these fonts, character sets are incomplete, random glyphs look wrong, critical features are missing, spacing is sloppy, and styles like bold and italic are either nonexistent or incongruous with the main style
- Design pros will immediately spot this :-/
- Still: in your exercise you are allowed to use fonts.google.com or download fonts from a free font platform

Getting webfonts

- Fonts you select at Adobe Typekit are added to a "kit" that you publish and connect to your web project. Because Typekit hosts the fonts and offers a standard licensing agreement, there’s not really anything you need to "get."
- Google fonts also offers an easy way to link to the selected fonts
- If you download or purchase a webfont, you will get the files: WOFF, or preferably WOFF2, as well as a copy of your End User License Agreement (EULA) specifying that web use is ok

Using webfonts

- You will link this cloud fonts in the head part of your HTML file, and these services will tell you how
- You will need to add the downloaded fonts to your css file
- The CSS @font-face rule (font files to stylesheets by defining fonts like so:

```
@font-face {
    font-family: "Source Sans Pro";
    font-weight: 400;
    font-style: normal;
    font-stretch: normal;
    src: url("SourceSansPro-Regular.woff2") format("woff2");
}
```

Here's an example using the "deepest possible browser support" recommendation from CSS-Tricks ( ): https://css-tricks.com/snippets/css/using-font-face/

Declaring the fonts

Now that your font files are online and defined in your CSS, you can use them in declarations like this:

```
p {
    font-family: "Source Sans Pro";
    font-size: 1.5rem;
}

p strong {
    font-style: bold;
}
```

Because you've defined the Source Sans Pro family, the browser knows which files to use when you refer to that name in a declaration. If you're writing your own @font-face definitions, the family name can be anything you wish. If you're using Typekit or Google fonts, check the Kit Editor for the family name.

Designing your page

Creating text-based layouts

- Everything matters here: the fonts you select (do they match?), the horizontal margins, the vertical margins, the line heights, …
- Everything needs to be in balance
- There is no shortcut. People often turn to script libraries or frameworks they hope will make it easier to manage a site’s typography or save development time by handling all of the details
- Such a typographic system makes decisions on your behalf. But in the end it al comes down to the skills of the designer. You need to be able to do it yourself

Limiting the width of your text

```
:root {
    font-size: 62.5%;}

body {
    font-family: sans-serif;
    line-height: 1.5rem;
    font-size: 1.3rem;}

main {
    width: 40rem; /* or max-width */}
```

Limiting the width of your text

Of course CSS Grid Layout can also take care of this for you

```
:root {
    font-size: 62.5%;}

body {
    font-family: sans-serif;
    line-height: 1.5rem;
    font-size: 1.3rem;}

main {
    display: grid;
    grid-template-columns: 1fr 1fr 1fr;
    max-width: 80rem;}
```

Limiting the width of your text

- For the exercise I would recommend you also experiment with vw and vh
- 100vw or viewport width equals the available width of the viewport, 50vw half of that, ...
- 100vh or viewport width equals the available width of the viewport, 50vh half of that, ...
- You also have vmin (the smallest, be it vw or vh) and vmax (the largest, be it vw or vh)

```
body {
    width: 70vw;
    height: 100vh;}

main {
    width: 50vmin;
    height: 40vmax;}
```

Adjust line-spacing accordingly

Adjusting line-height in your css can make or break a design. It is really important and something you need to experiment with

```
p {
    font-size: 1.5rem;
    line-height: 3rem;
}
```

Vertical margins

- If you're using a margin between paragraphs, try a margin-bottom equivalent to half of the line height, or try a small number from your modular scale
- Whichever value you choose, use rem units so that vertical margins relate to the body-text font size
- And set vertical margins in a single direction on every element: https://csswizardry.com/2012/06/single-direction-margin-declarations/

```
h1 {
    margin-bottom: 3rem;}

p {
    margin-bottom: 1.5rem;}
```

Horizontal margins

- Here also, you can work on feeling
- Horizontal margin sizes can fluctuate a lot, from very tight to very loose, and still look okay
- But make deliberate decisions about these, too. At the body text's narrowest, or when a small screen pressures the text to be even narrower, what amount of marginal, horizontal space should surround the text?
- Here of course CSS Grid comes into play (or positioning)

Different sizes for different screen widths

Don't forget that by using @media queries, you can change the font sizes of your designs

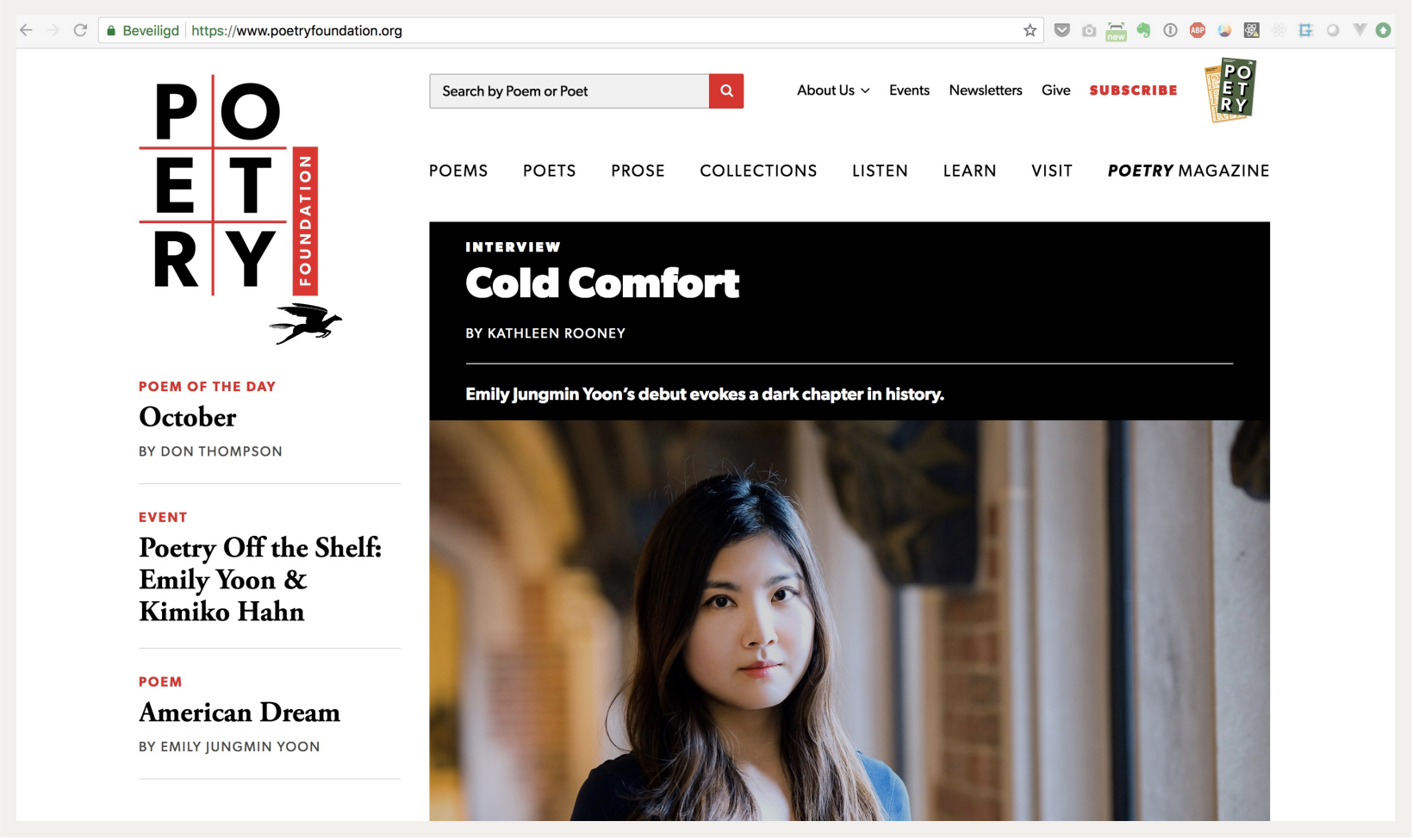
```
:root {
    font-size: 70%;
}

@media screen and (min-width: 70rem) {
        :root {
            font-size: 62.5%;
        }
}
```

Your exercise for this week

Exercise

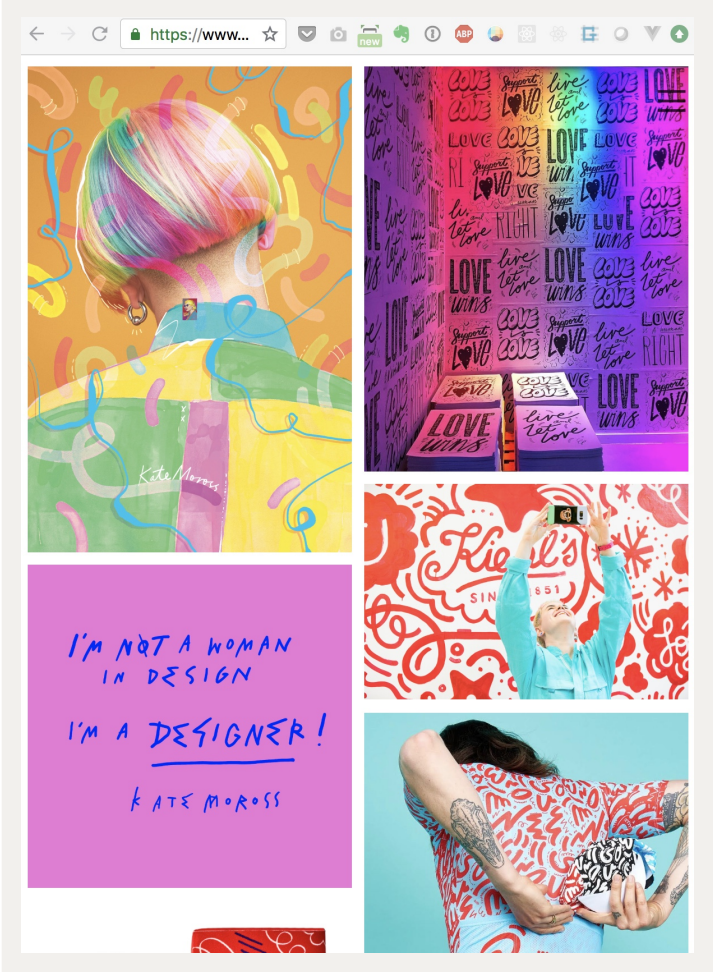
- You will make a web page combining 5 poems you have selected from http://www.poetryfoundation.org
- You will select at least 3 webfonts that you will use appropriately on this page
- Pay attention to all the stuff we have seen today: all that stuff matters
- Choose your heights, margins, line-heights, colors, ... carefully
- Your site needs to be typography (and mobile) first, and should be responsive
- Please take note that this is a visual experiment, not an actual website that you are making
- As a tip: look at the site of London based designer Kate Moross: https://www.katemoross.com/ For smaller screen widths the images are stacked underneath each other. For larger screen widths position is used to give this a spread effect. Perhaps you could do the same with your poems?

Poetry Foundation



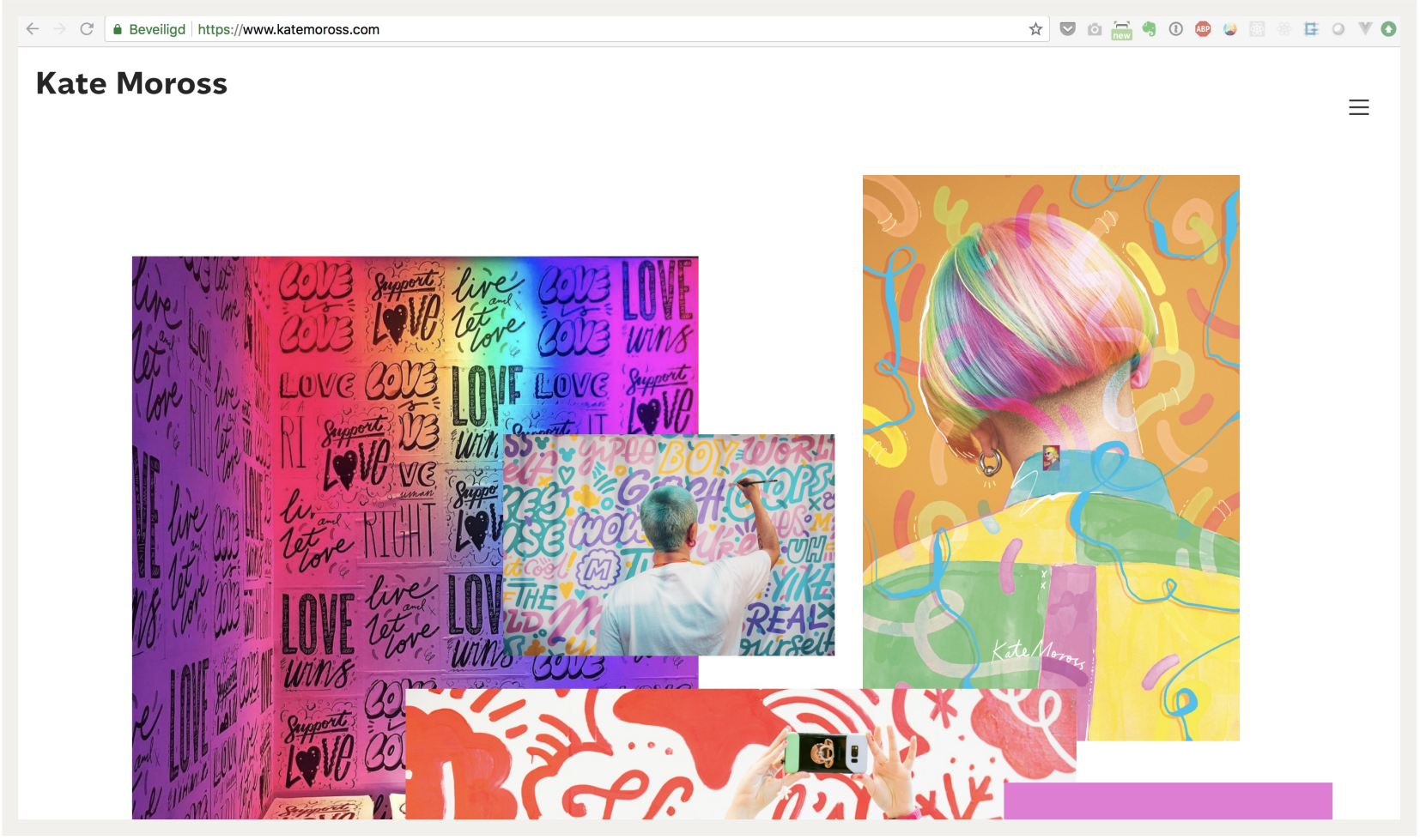
https://www.poetryfoundation.org

Kate Moross



https://katemoross.com

Kate Moross



https://katemoross.com

Information Technology 3

Questions?

kristof.michiels01@ap.be