LEARN TYPOGRAPHY BY BUILDING A NUTRITION LABEL

Introduction:

Typography is the art of styling your text to be easily readable and suit its purpose.

In this course, you'll use typography to build a nutrition label webpage. You'll learn how to style text, adjust line height, and position your text using CSS.

Step 1:

We've provided a basic HTML boilerplate for you.

Create an `h1` element within your `body` element and give it the text `Nutrition Facts`.

Step 2:

Below your `h1` element, add a `p` element with the text `8 servings per container`.

Step 3:

Add a second `p` element with the text `Serving size 2/3 cup (55g)`.

Step 4:

Within your `head` element, add a `link` element with the `rel` attribute set to `stylesheet` and the `href` attribute set to `https://fonts.googleapis.com/css?family=Open+Sans:400,700,800`.

This will import the Open Sans font family, with the font weight values 400, 700, and 800.

Also add a link element to link your styles.css file.

Step 5:

Create a body selector and give it a font-family set to Open Sans with a fallback of sans-serif.

Remember that fonts with spaces in the name must be wrapped in quotes for CSS.

Step 6:

The font is a bit small. Create an html selector and set the font to have a size of 16px.

Step 7:

Wrap your h1 and p elements in a div element. Give that div a class attribute set to label.

Step 8:

Borders can be used to group and prioritize content.

Create a .label selector and give it a border set to 2px solid black.

Step 9:

Good use of white space can bring focus to the important elements of your page, and help guide your user's eyes through your text.

Give your .label selector a width property set to 270px.

Step 10:

Give your .label selector a margin property set to 20px auto, and a padding property set to 0 7px.

Step 11:

If you inspect your .label element with your browser's developer tools, you may notice that it's actually 288 pixels wide instead of 270. This is because, by default, the browser includes the border and padding when determining an element's size.

To solve this, reset the box model by creating a * selector and giving it a box-sizing property of border-box.

Step 12:

Remember that the use of h1, h2, and similar tags determine the semantic structure of your HTML. However, you can adjust the CSS of these elements to control the visual flow and hierarchy.

Create an h1 rule and set the font-weight property to 800. This will make your h1 text bolder.

Step 13:

Give your h1 selector a text-align property of center.

Step 14:

Fine-tune the placement of your h1 by giving it a top and bottom margin of -4px and a left and right margin of 0.

Step 15:

Create a p selector and remove all margins.

Step 16:

Lines can help separate and group important content, especially when space is limited.

Create a div element below your h1 element, and give it a class attribute set to divider.

Step 17:

Create a selector for your new .divider and set the border-bottom property to 1px solid #888989. Also give it a top and bottom margin of 2px. It should not have any left or right margin.

Step 18:

The letter-spacing property can be used to adjust the space between each character of text in an element.

Give your h1 selector a letter-spacing property set to 0.15px to space them out a bit more.

Step 19:

Nutrition labels have a lot of bold text to draw attention to important information. Rather than targeting each element that needs to be bold, it is more efficient to use a class to apply the bold styling to every element.

Give your second `p` element a `class` attribute set to `bold`.

Step 20:

Your new class does not have any styling yet. Create a `.bold` selector and give it a `font-weight` property set to `800` to make the text bold.

Go ahead and remove the `font-weight` property from your `h1` selector as well.

Step 21:

Give your `h1` element a `class` attribute set to `bold`. This will make the text bold again.

Step 22:

Horizontal spacing between equally important elements can increase the readability of your text.

Wrap the text `2/3 cup (55g)` in a `span` element.

Step 23:

Now we can add the horizontal spacing using `flex`. In your `p` selector, add a `display` property set to `flex` and a `justify-content` property set to `space-between`.

Step 24:

Wrap everything within the `.label` element in a new `header` element.

Step 25:

Now update your `h1` selector to be `header h1` to specifically target your `h1` element within your new `header`.

Step 26:

Create a new `div` element below your `header` element, and give it a `class` attribute set to `divider large`.

Step 27:

Create a new `.large` selector and give it a `height` property set to `10px`. Also create an `.large, .medium` selector and set the `background-color` property to `black`.

Step 28:

You may notice there is still a small border at the bottom of your `.large` element. To reset this, give your `.large, .medium` selector a `border` property set to `0`.

Note: the `medium`(medium) class will be utilized later for the thinner bars of the nutrition label.

Step 29:

Create a new `div` below your `.large` element and give it a `class` attribute set to `calories-info`.

Step 30:

Within your .calories-info element, create a div element. Give that div element a class attribute set to left-container. Within the newly created div element, create a h2 element with the text Amount per serving. Give the h2 element a class attribute set to bold small-text.

Step 31:

The rem unit stands for root em, and is relative to the font size of the html element.

Create a .small-text selector and set the font-size to 0.85rem, which would calculate to roughly 13.6px (remember that you set your html to have a font-size of 16px).

Step 32:

Create a .calories-info h2 selector and remove all margins.

Step 33:

Below your .small-text element, create a new p element with the text Calories. Also below the .left-container element, create a new span element with the text 230.

Step 34:

Create a new .calories-info selector and give it a display property set to flex. Also give it a justify-content property set to space-between and align-items property set to flex-end.

Step 35:

Create a new `.left-container p` selector setting the top and bottom margin to `-5px`, and the left and right margin to `-2px`. Also set the `font-size` to `2em` and `font-weight` to `700`.

Step 36:

Create a `.calories-info span` selector, set its `font-size` to `2.4em` and `font-weight` to `700`.

Step 37:

Typography is often more art than science. You may have to tweak things like alignment until it looks correct.

Give your `.calories-info span` selector a `margin` set to `-7px -2px`. This will shift your `230` text into place.

Step 38:

Below your `.calories-info` element, add a `div` with the `class` attribute set to `divider medium`.

Step 39:

Create an `.medium` selector and give it a `height` property of `5px`.

Step 40:

Create a new `div` element below your `.medium` element. Give it a `class` attribute set to `daily-value small-text`. Within this new `div`, add a `p` element with the text `% Daily Value *`, and set the `class` attribute to `bold right`.

Step 41:

The text `% Daily Value *` should be aligned to the right. Create a `.right` selector and use the `justify-content` property to do it.

Step 42:

Use your existing `.divider` element as an example to add a new divider after the `p` element.

Step 43:

After your last `.divider` element, create a `p` element and give it the text `Total Fat 8g 10%`. Then, wrap the text `Total Fat` in one `span` element, the text `10%` in another, and give them each a class of `bold`.

Step 44:

Notice how the text `8g` appears centered in the preview. Nest the `span` element containing the text `Total Fat` along with the text `8g`, in an additional `span` element for alignment.

Step 45:

Below your element with the `Total Fat` text, create a new `p` element with the text `Saturated Fat 1g 5%`. Wrap the `5%` in a `span` with the

class attribute set to bold. In this case this is enough to align the percentage to 5%.

Step 46:

This new p element will need to be indented. Give it a class set to indent.

Step 47:

Create a new .indent selector and give it a margin-left property set to 1em.

Step 48:

Create a .daily-value p selector to target all of your p elements in the daily-value section. Give this new selector a border-bottom set to 1px solid #888989.

Step 49:

The bottom borders under your % Daily Value * and Saturated Fat 1g 5% elements do not extend the full width of the label. Add no-divider to the class for these two elements.

Step 50:

The :not pseudo-selector can be used to select all elements that do not match the given CSS rule.

Example Code:

```
div:not(#example) {
  color: red;
}
```

The above selects all `div` elements without an `id` of `example`.

Modify your `.daily-value p` selector to exclude the `.no-divider` elements.

Step 51:

Now you will have to add separate dividers below your `.no-divider` elements.

Your first `.no-divider` element has a `.divider` after it. Create another `.divider` after your second `.no-divider` element.

Step 52:

After your last `.divider`, create another `p` element with the text `Trans Fat 0g`. Italicize the word `Trans` by wrapping it in an `i` element. Give the new `p` element the `class` attribute set to `indent no-divider`. Wrap `Trans Fat 0g` in a `span` element for alignment.

Step 53:

Create another `.divider` after your last `p` element.

Step 54:

After your last `.divider`, create a new `p` element with the text `Cholesterol 0mg 0%`. Then, wrap the text `Cholesterol` in a `span` element, `0%` in another, and give each of them a class of `bold`.

Finally, nest the span element containing the text Cholesterol along with the text 0mg, in an additional span element for alignment.

Step 55:

Below your last p element, create another p element with the text Sodium 160mg 7%. Put Sodium and 7% each in their own span with a class of a bold like you did with the others.

Then, add an additional span element around Sodium 160mg for alignment again.

Step 56:

Below your last p element, add another p element with the text Total Carbohydrate 37g 13%. Like before, use span elements to make the text Total Carbohydrate and 13% bold. Then, wrap the nutrient and amount in a span for alignment again.

Step 57:

Below your last p element, add another p element with the text Dietary Fiber 4g. Give the p element the class necessary to indent it and remove the dividing border. Then create a divider below that p element.

Step 58:

Create another p element after your last .divider, and give it the text Total Sugars 12g. Assign that p element the class values necessary to indent it and remove the bottom border. Then create another .divider below your new p element.

Step 59:

The advantage to creating these dividers is that you can apply specific classes to style them individually. Add `double-indent` to the `class` for your last `.divider`.

Step 60:

Create a `.double-indent` selector and give it a left margin of `2em.`

Step 61:

Below your `.double-indent` element, add a new `p` element with the text `Includes 10g Added Sugars 20%`. Your new `p` element should also be double indented, and have no bottom border. Use a `span` to make the `20%` bold and right aligned.

Then create another divider after that `p` element.

Step 62:

After your last divider, create another `p` element with the text `Protein 3g`. Use the necessary classes to remove the bottom border, and a `span` to make the `Protein` bold. Then wrap the text `Protein 3g` including the new `span` element, in a new `span` element.

Following this element, create a large divider.

Step 63:

Create another `p` element below your large divider. Give the `p` element the text `Vitamin D 2mcg 10%`.

The p element contains only text, you can wrap the percentage in a span element so that it is considered a separate entity from the rest of the text, and it's moved to the right.

Step 64:

Create another p element, give it the text Calcium 260mg 20%. Align 20% to the right. Below that, create a p element with the text Iron 8mg 45%, aligning the 45% to the right.

Step 65:

Create the final p element for your .daily-value section. Give it the text Potassium 235mg 6%. Align the 6% text to the right, and remove the bottom border of the p element.

Step 66:

Add a medium divider after your .daily-value element. Below that new divider, create a p element with the class attribute set to note.

Give the p element the following text:

Example Code:

* The % Daily Value (DV) tells you how much a nutrient in a serving of food contributes to a daily diet. 2,000 calories a day is used for general nutrition advice.

Step 67:

Create a .note selector, and set the size of the font to 0.6rem. Also set the top and bottom margins to 5px, removing the left and right margins.

Step 68:

Give the `.note` selector a left and right padding of `8px`, removing the top and bottom padding. Also set the `text-indent` property to `-8px`.

With these last changes, your nutrition label is complete!