Responsive Web Design

<meta name="viewport" content="width=device-width, initial-scale=1, minimal-ui, user-scalable=no">

many relative measurements. Use **rem** for **font-size;**
use a xmls file for a SVG generated image(using GIMP+PHOTOSHOP to transfer a png into a svg)

Today, the **em** represents the size of the base font being used. For example, if the base font of a browser is **16 pixels** (which is normally the default size of text in a browser), then 1 em is equal to 16 pixels. **2 ems would equal 32 pixels**, and so on.
Let's take a look at two examples that show how em can be used in CSS.

```
.heading {
  font-size: 2em; //=32px
}
```

In the example above, no base font has been specified, therefore the font size of the heading element will be set relative to the default font size of the browser. Assuming the default font size is 16 pixels, then the font size of the heading element will be 32 pixels.

```
.splash-section {
  font-size: 18px;
}

.splash-section h1 {
  font-size: 1.5em;
}
```

The example above shows how to **use ems without relying on the default font size of the browser. Instead, a base font size (18px)** is defined for all text within the splash-section element. The second CSS rule will set the font size of all h1 elements inside of splash-section relative to the base font of splash-section (18 pixels). The resulting font size of h1 elements will be 27 pixels.

rem stands for *root em*. It acts similar to em, but instead of checking parent elements to size font, it checks the *root element*. **The root elemrement is the <html> tag.**
One advantage of using rems is that all elements are compared to the same font size value, making it easy to predict how large or small font will appear. If you are interested in sizing elements consistently across an entire website, the rem measurement is the best unit for the job. If you're interested in sizing elements in comparison to other elements nearby, then the em unit would be better suited for the job.

To size non-text HTML elements relative to their parent elements on the page you can use *percentages*.
Be careful, a child element's dimensions may be set erroneously if the dimensions of its parent element aren't set first.

For example, when a property like **margin-left** is set using a percentage (say 50%), the element will **be moved** halfway **to the right in the parent container** (as opposed to the child element receiving a margin half of its parent's margin).

When height and width are set using percentages, you learned that the dimensions of child elements are calculated based on the dimensions of the parent element.
When percentages are used to set **padding and margin**, however, **they are calculated based only on the *width* of the parent element**.
For example, when a property like margin-left is set using a percentage (say 50%), the element will be moved halfway to the right in the parent container (as opposed to the child element receiving a margin half of its parent's margin).
Vertical padding and margin are also calculated based on the width of the parent. Why? Consider the following scenario:

1. A container div is defined, but its height is not set (meaning it's flat).
2. The container then has a child element added within. The child element *does* have a set height. This causes the height of its parent container to stretch to that height.
3. The child element requires a change, and its height is modified. This causes the parent container's height to also stretch to the new height. This cycle occurs endlessly whenever the child element's height is changed!

In the scenario above, an unset height (the parent's) results in a constantly changing height due to changes to the child element. This is why vertical padding and margin are based on the width of the parent, and not the height.
Note: When using relative sizing, ems and rems should be used to size text and dimensions on the page related to text size (i.e. padding around text). This creates a consistent layout based on text size. Otherwise, percentages should be used.

Let's size the height of the banner relative to the root element's font size.
In **style.css**, set the height in #banner to 46rem.
Note: The root element's font size is 16 pixels, meaning that 46rem will result in a height of 736 pixels.

Although relative measurements provide consistent layouts across devices of different screen sizes, elements on a website can lose their integrity when they become too small or large. You can limit how wide an element becomes like this:

1.

```
p {
  min-width: 300px;
  max-width: 600px;
}
```
In the example above, when the browser is resized, the width of paragraph elements will not fall below 300 pixels, nor will their width exceed 600 pixels.
Note: The unit of pixels is used to ensure hard limits on the dimensions of the element(s).

```
.container {
  width: 50%;
  height: 200px;
  overflow: hidden;
}

.container img {
  max-width: 100%;
  height: auto;
  display: block; //instruct the images to behave as block-level elements and facilitate scaling (as opposed to their default inline behavior
}
```
In the example above, .container represents a container div. It is set to a width of 50% (half of the browser's width, in this example) and a height of 200 pixels. Setting overflow to hidden ensures that any content with dimensions larger than the container will be hidden from view.
The second CSS rule ensures that images scale with the width of the container. The height property is set to auto, meaning an image's height will *automatically* scale proportionally with the width. Finally, the last line will display images as block level elements (rather than inline-block, their default state). This will prevent images from attempting to align with other content on the page (like text), which can add unintended margin to the images.
It's worth memorizing the entire example above. It represents a *very common* design pattern used to scale images and videos proportionally.
Note: The example above scales the width of an image (or video) to the width of a container. If the image is larger than the container, the vertical portion of the image will overflow and will not display. **To swap this behavior, you can set max-height to 100% and width to auto (essentially swapping the values).** This will scale the *height* of the image with the height of the container instead. If the image is larger than the container, the horizontal portion of the image will overflow and not display.

Background images of HTML elements can also be scaled responsively using CSS properties.

```
body {
  background-image: url('#');
  background-repeat: no-repeat;
  background-position: center;
  background-size: cover;
}
```

In the example above, the first CSS declaration sets the background image (# is a placeholder for an image URL in this example). The second declaration instructs the CSS compiler to not repeat the image (by default, images will repeat). The third declaration centers the image within the element.

The final declaration, however, is the focus of the example above. It's what scales the background image. The image will *cover* the entire background of the element, all while keeping the image in proportion. If the dimensions of the image exceed the dimensions of the container then only a portion of the image will display.

- Content on a website can be sized relative to other elements on the page using *relative measurements*.
- The unit of em sizes font relative to the font size of a parent element.
- The unit of rem sizes font relative to the font size of a root element. That root element is the <html> element.
- Percentages are commonly used to size box-model features, like the width, height, borders, padding, or margin of an element.
- When percentages are used to size width and height, child elements will be sized relative to the dimensions of their parent (remember that parent dimensions must first be set).
- Percentages can be used to set padding and margin. Horizontal and vertical padding and margin are set relative to the width of a parent element.
- The minimum and maximum width of elements can be set using min-width and max-width.
- The minimum and maximum height of elements can be set using min-height and max-height.
- When the height of an image or video is set, then its width can be set to auto so that the media scales proportionally. Reversing these two properties and values will also achieve the same result.
- A background image of an HTML element will scale proportionally when its background-size property is set to cover.