HTML and CSS

Table Of Content

Module 1 - Getting Started

Create Your First HTML Page

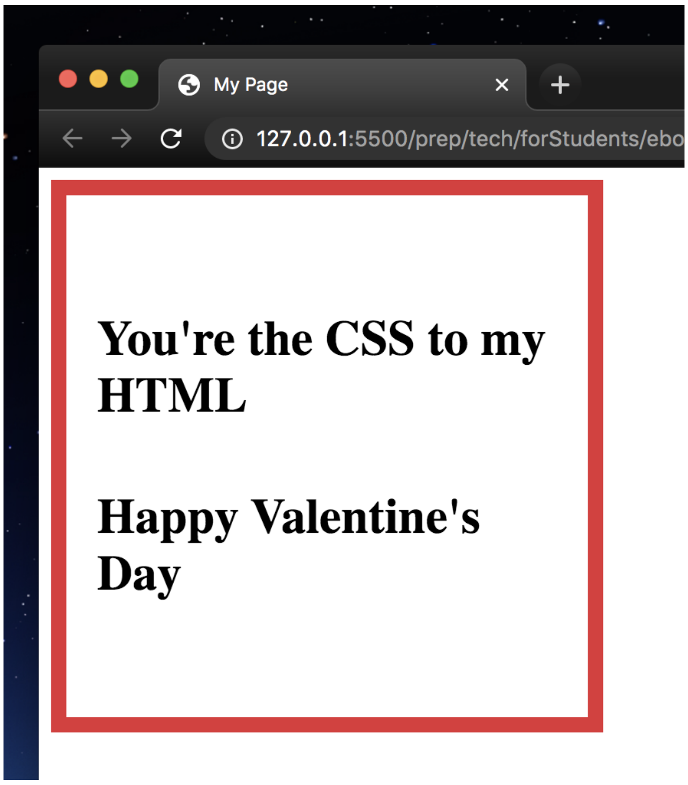
- Create an `app` folder on your computer
- Create file named `index.html` inside `app` folder on your computer
- Open that page in your favorite text editor
    - I personally use VSCode

Module 2 - Styling your Webpage

Now let's add your first style

- Let's add styles for your valentine's day card
- We are using `.card` - class selector to grab the card DIV and style it
- Here we are just setting a nice red `border: 10px solid #E53038;`
- `height: 100vh;` is done to match out `body` tag's height - which is the full view-port height.

- `display: flex;` makes this `card` DIV a flex-box.
    - We are just making all our `flex-children` align in vertically and horizontally center position in one column.
    - NOTE: We will learn about flex-box in the later section.



You're the CSS to my HTML

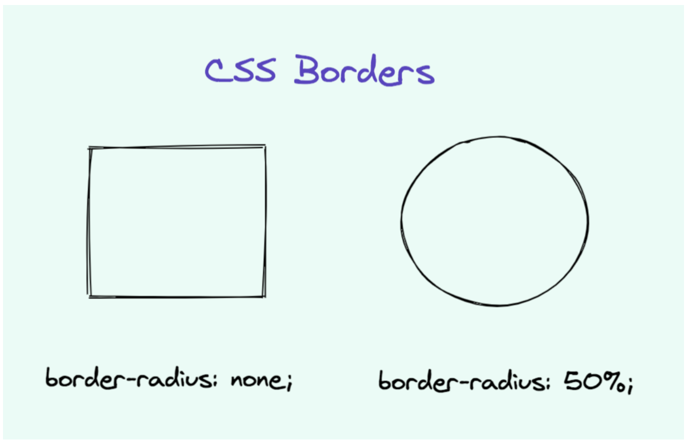
Happy Valentine's Day

```css
.card {
  border: 10px solid #E53038;
  height: 300px;
  width: 300px;
  padding: 20px;
  display: flex;
  flex-direction: column;
  justify-content: center;
  align-items: center;
}
```

Fun with Border Radius

Shapes

- Borders also take another property called `border-radius` using which you can give different shapes to your elements

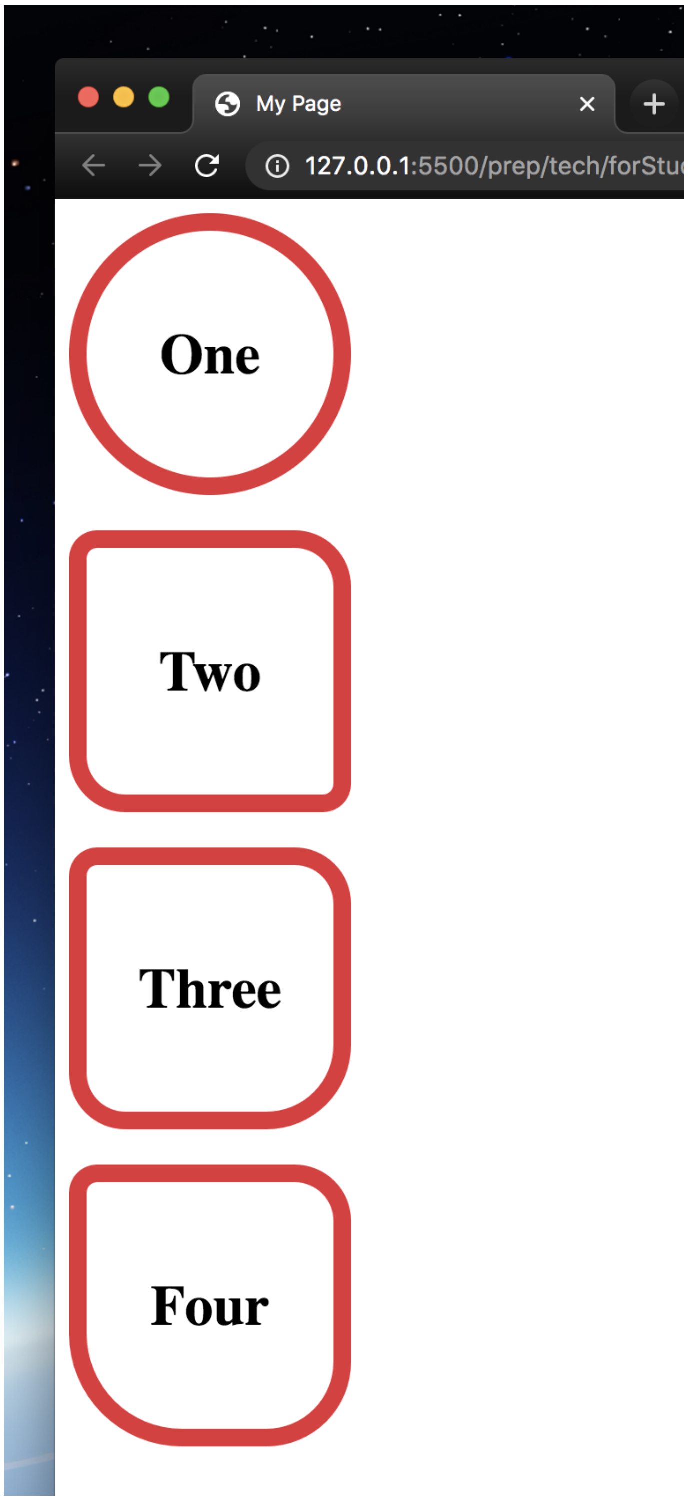


- In the above illustration we have a square on the left and circle on the right
- If you provide `border-radius` of `50%` it will turn your square into a circle

```
.square {
  border-radius: none;
}

.circle {
  border-radius: 50%;
}
```

Shorthand

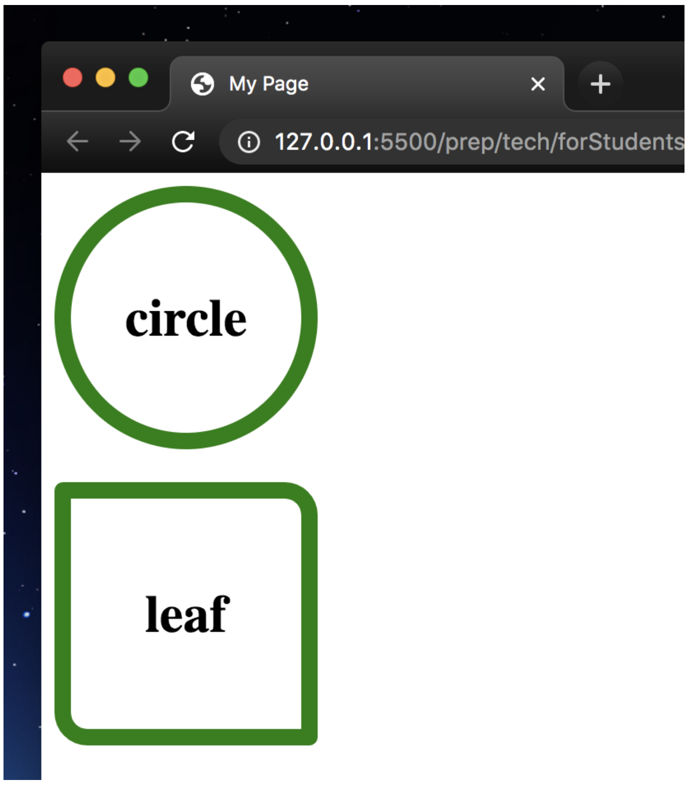
- If `one` value is set, this radius applies to all 4 corners.
- If `two` values are set, the first applies to `top-left and bottom-right` corner, the second applies to `top-right and bottom-left` corner.
- If `three` values are set - the `second` value applies to `top-right and also bottom-left`.
- If `four` values apply to the `top-left, top-right, bottom-right, bottom-left` corner (in that order).



```
<div class="card one">
  <h1 class="">One</h1>
</div>
<div class="card two">
  <h1 class="">Two</h1>
</div>
<div class="card three">
  <h1 class="">Three</h1>
</div>
<div class="card four">
  <h1 class="">Four</h1>
</div>
```

```
// all 4 corners
.one {
  border-radius: 50%;
}

// 10% top-left and bottom-right,  20% top-right and bottom-left
.two {
  border-radius: 10% 20%
}

// 10% top-left, 20% top-right and also bottom-left, 30% bottom-right
.three {
  border-radius: 10% 20% 30%;
}

// top-left, top-right, bottom-right, bottom-left corner (in that order)
.four {
  border-radius: 10% 20% 30% 40%;
}

.card {
  border: 10px solid #E53038;
  height: 100px;
  width: 100px;
  padding: 20px;
  display: flex;
  flex-direction: column;
  justify-content: center;
  align-items: center;
  margin-bottom: 20px;
}
```

Circle and leaf



Circle

```
.circle {
    border-radius: 50%;
}
```

Leaf

```
.leaf {
    border-radius: 5px 20px 5px;
}
```

Module 3 - Display and position your elements

Paddings

- Paddings are used to generate space around the given element's content - inside its border

```
<div class="myDiv">
  <p>My Paragraph</p>
</div>

// styles

.myDiv {
  padding: 20px;
}
```

- `padding: 20px;` gives the `div` element padding of `20px`
- So, basically there will be `20px` space between `p` and `div` on all the sides

Padding On Individual Sides

- You can also give padding to the elements on any particular side if you'd want

```
padding-top
padding-right
padding-bottom
padding-left
```

Padding Shorthands

- To give padding on all the sides

```
div {
  padding: 20px;
}
```

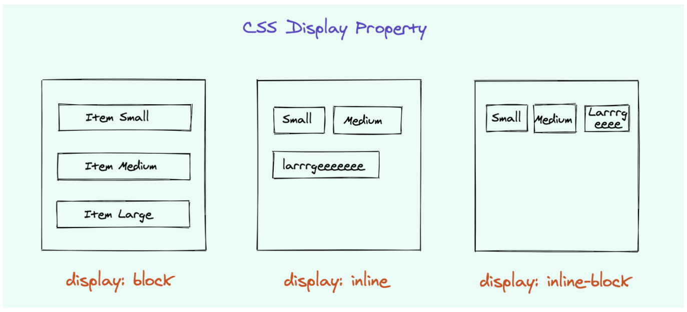
- The below example give padding `20px` top and bottom
- And give padding `40px` left and right

```
div {
  padding: 20px 40px;
}
```

- The below example give padding `20px` top
- And give padding `40px` left and right
- And give padding `50px` bottom

```
div {
  padding: 20px 40px 50px;
}
```

Display



Block

- This property stretches the element left to right as far as it can
- Default is block for `div, p, form, header, footer, section` (and some more)
- Such elements cannot be placed on the same horizontal line with any other display modes
  - Except when they are floated
- Like shown in the illustration -> every item stretches and uses up the entire row

Inline

- Inline element sits in line
  - Without disrupting the flow of other elements
- Like shown in the illustration -> every item takes up only the space it needs
  - Item wraps to the next row if there is no enough space
- `span, em, b` are examples of inline elements
- They take only width needed for it
- They do not honor vertical padding
  - No width
  - No height
  - They just ignores them
- Horizontal margin and padding are honored
- Vertical margin and padding are ignored

Inline-block

- This is just like inline element
- BUT they will respect the width and height
- Basically, they combine the properties of both block elements and inline elements
- The element can appear on the same horizontal line as other elements
- So, like the illustration shows if you set width you can fit all the items together in a single row

None

- These elements will not appear on the page at all
- But you can still interact with it through DOM
- NO space allocated on the page

Visibility Hidden

- Space is allocated for it on the page
- Tag is rendered on the DOM
- The element is just no visible

```
div {
  visibility: hidden;
}
```

Flex

- Flex property gives ability to alter its item's width/height to best fill the available space
- It is used to accommodate all kind of display devices and screen sizes
- Fills available space
    - Or shrink to prevent overflow

Module 4 - Semantic HTML5

What is HTML5 Web Storage?

- With HTML5, browsers can store data locally
- It is more secure and faster than cookies
- You are able to store large information -> more than cookies
- They are `name/value` pairs
- 2 objects
  - `window.localStorage` - stores data with no expiration date
  - `window.sessionStorage` - stores data for one session (data is lost when the tab is closed)

localStorage:

- Stores the data with no expiration date
- Data is NOT deleted when browser is closed
- Available the next day, week, or year
  - It's not possible to specify expiration
  - You can manage its expiration in your app

```
// Store
localStorage.setItem("lastname", "Smith");

// Retrieve
document.getElementById("result").innerHTML =
localStorage.getItem("lastname");
```

sessionStorage:

- Stores the data for only one session
- Data deleted when browser is closed

```
if (sessionStorage.clickcount) {
    sessionStorage.clickcount = Number(sessionStorage.clickcount) + 1;
} else {
    sessionStorage.clickcount = 1;
}

document.getElementById("result").innerHTML = "You have clicked the button
" +
sessionStorage.clickcount + " time(s) in this session.";
```

Module 5 - Flexbox intro and media query

Let's talk about the sizes - `px` vs `em` vs `rem`

- `Pixels` are ignorant, avoid using them
- If you're setting all of your font-sizes, element sizes and spacing in pixels, you're not treating the end user with respect.
  - Users will have to zoom in and out with ctrl plus +/- depending on the device they are on
- `REMs` are a way of setting font-sizes based on the font-size of the root HTML element
- Allows you to quickly scale an entire project by changing the root font-size
- `em` is relative to the font size of its direct or nearest parent
  - When you have nested styles it becomes difficult to track `ems`
  - This is what REMs solve - the size always refers back to the root
- Both `pixels` and `REMs` for media queries fail in various browsers when using browser zoom, and `EMs` are the best option we have.

How to calculate PX from REM

EX: HTML font-size is set to 10px, paragraph font-size is set to 1.6rem

Then size in pixels is - `1.6rem * 10px = 16px`

Module 6 - Quirks, tips, and tricks

Normalizing CSS

- Small CSS file that provides cross-browser consistency
- Provides default styles for HTML elements
- Make sure that all HTML elements renders the same way in ALL browsers - same padding, margin, border, etc..
- In some cases this approach applies IE or EDGE styles to the rest of the browsers

Reset CSS

- This approach says that we don't need the browsers' default styles at all
- We'll define in the project according to our needs
- `CSS Reset` resets all of the styles that come with the browser's user agent
- Grab sample CSS reset here
- The problem with CSS Resets is that they are ugly and hard to debug
- Solution - use Normalize CSS with little bit of CSS Reset
- Unlike an ordinary CSS reset, target specific HTML tags' styles rather than making a big list of tags.
    - Make it less aggressive and a lot more readable