colorga.me

Learning modern front-end development, by example

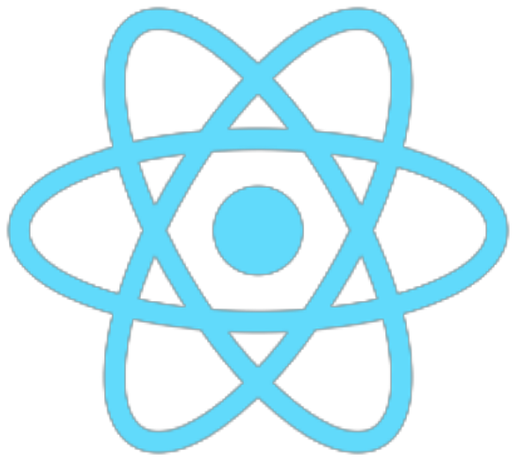
10 years ago….

- Javascript was useless, and (mostly) despised
- CSS3 was brand-new with very little features
- Most websites were…
  - Static
  - Written in HTML, CSS with a touch of vanilla JS if required

What do we need today?

- Responsive websites
  - We have phones and tablets with weird aspect ratios now oops
- Progressive Web Apps (PWAs)
- Complex Single Page Applications (SPAs)
- NativeScript, React Native (Javascript on Mobile)


BABEL




NUXT


Sass

OK, slowing down…

- Frameworks
  - Front-end frameworks *(Bootstrap, MaterialCSS)*
  - Javascript frameworks *(VueJS, Angular, React)*
- Tools
  - Transpiler tools *(Babel, Sass)*
  - Application bundler *(Webpack, Parcel)*
  - Package managers *(Yarn, NPM)*
- Framework Tools
  - NativeScript, React Native, Electron

Let’s get started…

colorga.me

1

2

3

Simple color game :)

- Select the color hue box that matches the background to gain points

- Counter below increments with the points

- Becomes harder to differentiate as time progresses

- Fancy animations

- Fully responsive

Chapter 1:

Laying the HTML bricks

Create HTML file

- Create a new directory for the project *[mkdir colorgame]*
- Create a index.html **file***[vim index.html]*
- Insert following snippet

```
<!doctype html>
<html lang="en">
  <head>
    <!-- Required meta tags -->
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width">

    <link href="https://fonts.googleapis.com/css?family=Source+Sans+Pro:
300,700" rel="stylesheet">
    <title>colorga.me</title>
  </head>
  <body>
    <div class="container">
      <h1>colorga.me</h1>
      <ul>
        <li><button type="button" class="option" id="one">1</button></li>
        <li><button type="button" class="option" id="two">2</button></li>
        <li><button type="button" class="option" id="three">3</button></li>
      </ul>
      <span id="counter">0</span>
    </div>

    <script src="./index.js"></script>
  </body>
</html>
```

Chapter 2:

Time to be lazy

Yarn, package manager

- Even front-end projects use other node modules
  - Need a package manager to manage the installation of modules
  - Using a random color generator in the project
- Want to be lazy
  - Install application bundling tools (coming soon)
- *yarn init*

Parcel, the application bundler

- Blazing fast, zero-conf application bundler
  - Better than webpack for small projects, requires less time
- *yarn add parcel-bundler -D*
- *parcel index.html*
- Add the command to package.json
  - “serve”: “parcel -p 3000 index.html”
- *yarn serve*

Let’s install more tooling

- *yarn add normalize.css*
- *yarn add autoprefixer -D*
- *yarn add babel-cli babel-preset-env -D*
- Create .babelrc and .postcssrc files
- Populate build script package.json
  - “build”: “parcel build index.html”

Chapter 3:

CSS is not magic

Create CSS file

- Create a main.css file *[vim main.css]*
- Do root and default element styling
  - Set :root font size to 16px
  - Set default font-family, size and weight settings
  - Set h1 element style
- CSS Units
  - vw, vh (viewport-width, viewport-height)
  - rem, em (root element, element)
  - % (percentage of parent element)

```css
:root {
  font-size: 16px;
}

body {
  font-family: 'Source Sans Pro', sans-serif;
  color: white;
  height: 100vh;
}

h1 {
  font-size: 2rem;
  font-weight: 700;
}
```

Styling overall layout

- Flexbox layout for overall position
- Set container height to fill up all of the parent element, *body*
- Set the box sizing to border-box to avoid padding making the element larger

```css
.container {
  box-sizing: border-box;

  display: flex;
  flex-direction: column;
  justify-content: space-between;
  align-items: center;

  background-color: #ff7f11;
  padding: 0rem 1rem 1rem 1rem;
  height: 100%;
}
```

Styling list

- Remove default list styling
- Use flex box to…
  - Align list elements in a row
  - Align to vertical and horizontal center
  - Remove default padding

```
.container ul {
  list-style: none;

  display: flex;
  flex-direction: row;
  justify-content: center;
  align-items: center;
  flex-wrap: wrap;

  padding: 0rem;
  margin: 0rem;
}
```

Styling each option

- Set to a fixed width and height
- Apply margins to separate each item
- Apply border-radius to add a fillet effect
- Set colors and create a border
- Enlarge on hover

```css
.option {
  box-sizing: border-box;

  width: 10rem;
  height: 10rem;
  margin: 2rem;
  font-size: 3rem;
  font-weight: 700;

  border: 0.6rem solid white;
  border-radius: 3rem;
  color: white;
  background-color: #ff7f11;

  transition: width 1s, height 1s;
}

.option:hover {
  width: 11rem;
  height: 11rem;

  transition: width 1s, height 1s;
}
```

Styling the counter

- Align counter to the end of the row flex box
- Set correct font-weight and size

```
#counter {
  align-self: flex-end;
  font-weight: 300;
  font-size: 3rem;
}
```

```
@media only screen and (max-width: 800px) {
  :root {
    font-size: 12px;
  }
}
```

Chapter 4:

JS is the real magic

https://github.com/solderneer/colorga.me/blob/master/index.js

Important points to note

- *yarn add randomcolor*
- randomcolor used to generate a random color (duh)
- *filter: hue-rotate(`${shift}deg`)* is used to define different color

Conclusion (ish)

Today I learnt...

- Yarn as a JS package manager
- ParcelJS as an application bundler
- Babel as a transpiler, autoprefixer and normalize for CSS transformation
- Basic HTML
- CSS3 magic
- ES6 module syntax

How to continue?

- Don’t dive into all the billions of plugins, extensions and tools headfirst
- Adopt frameworks and tools incrementally
- When you need something, you’ll find it

Thanks!

https://github.com/solderneer/colorga.me