Gala darbs

Kursa gala darba uzdevums ir, vadoties pēc figma dizaina faila, izstrādāt mājaslapu React ietvarā. Būtu ieteicams izstrādāt šo lapu, izveidojot 6-7 React komponentes. Bet šis skaits var atšķirties no katra individuālās pieejas uzdevumam.
Šajā PDF ar pseido koda piemēriem apraktītie risinājumi iespējamajiem izaicinājumiem. Dotās vērtības piemēros nav jāuztver kā pareizais risinājums, bet gan kā sintaktisks piemērs kā risināt šos izaicinājumus.
Viss kas saistīts ar media queries jeb nofromējuma pārslēgšanu no desktop uz mobile jātiek galā eksperimentējotu un vadoties pēc sākotnējo CSS nodarbību paraugiem vai citiem resursiem internetā.
Šeit nav aprkatīts risinājums katram elementam un komponentei pilnībā, bet gan tikai situācijas kuras neesam līdz šim risinājuši nodarbībās. Jebkurš cits aspekts par šīs lapas izstrādi ir nosegts nodarbību materiālos.
P.s Nāksies diez daudz izmantot **display: flex;**

1. Izmantojot Vite bibliotēku, izveidojam jaunu projektu. Termināli izpildīt komandu:
   npm create vite@latest

   Pēc tam sekot konfigurācijas soļiem:
   Project name: **vards-uzvards-galadarbs**
   Select a framework: **React**
   Select a variant: **JavaScript**

   Tad termināli izpildīt sekojošās komandas:
   cd vite-project
   npm install
   npm run dev

   Ja projektā plānojiet izmantot SVG failus, nepieciešams uzstādīt arī attiecīgo Vite plugin priekš šīs darbības:

   npm install vite-plugin-svgr

   Un konfigurēt Vite, jebpapildināt **vite.config.js** failu, lai šis plugin tiktu izmantots:

```
import { defineConfig } from "vite";
import react from "@vitejs/plugin-react";
import svgr from "vite-plugin-svgr";

export default defineConfig({
  plugins: [react(), svgr()],
});
```

Lai nodrošinātu SPA(Single Page Application) funkcionalitāti un varētu izstrādāt lapu navigāciju, nepieciešams uzstādīt **React Router**:

npm install react-router

Ja vēlamies izmantot “burger menu” no 10. Nodarbības lekcijas piemēra, nepieciešams to uzstādīt:

npm install react-burger-menu

2. Kad Vite un nepieciešamo npm paku uzstādīšanas process ir izdarīts, varam vadīties pēc Vite izveidotās mapju struktūras, pielāgot to pēc saviem ieskatiem vai sekot lekciju paraugam kur iekš **src** mapes mēs izveidojam mapes **components, pages, style** un **assets**.

3. Header komponentei vienīgās nianses ko ņemt vērtā:
   - Linku navigācija tiek aizvietota ar burger menu uz mobilo ierīču ekrāniem (768px)
   - Nepieciešams ievietot logo

   Nepieciesamo attēlu exportējam no Figma, ievietojam **assets** mapē un pārcaucam šo failu īsāk(piem., logo.png). Iekš jebkuras komponentes attēlus var ievietot šādi:

```
import Logo from "../../assets/logo.png";

const Header = () => {
  return (
    <img src={Logo} />
  )
};
```

4. Hero elementam, jeb pirmā readzamā komponente mājaslapā satur fona attēlu. Nepieciešamo attēlu eksportējam no Figma un ievietojam **assets** mapē.
   Fona attēlu jebkuram HTML elementam iestata iekš CSS šādi:

```
.hero {
  background-image: url("../../assets/hero-img.jpg");
  background-repeat: no-repeat;
  background-size: cover;
}
```

5. Iekš Hero elementa atrodas arī 2 pogas. Pogas ir komponente ar 2 dažādiem noformējumiem noformējumu. Viens no veidiem kā to panākt ir izmantojot komponentes props un atkarībā no padorā prop piešķirt citu CSS klasi jeb noformējumu pogai.

   Iekš **Hero** komponentes:

```jsx
const Hero = () => {
  return (
    <div className="hero">
      <Button text="Playstore" />
      <Button text="App store" isSecondary />
    </div>
  );
};
```

   Iekš **Button** komponentes:

```jsx
const Button = ({ text, isSecondary }) => {
  return (
    <button className={isSecondary ? "btn-secondary" : "btn-primary"}>
      {text}
    </button>
  );
};
```

6. “How the app works” komponentē izaicinājums ir pozicionēt attēlus. Attiecīgo attēlus nepieciešams exportēt no Figa PNG formātā, lai saglabātu caurspīdīgu fonu. Dizainā redzamo attēlu nobīdi varam panākt ar CSS transform īpašību:

```jsx
import PhoneLeft from "../../assets/phone-left.png";
import PhoneRight from "../../assets/phone-right.png";
import "./HowAppWorks.css";

const HowAppWorks = () => {
  return (
    <div className="how-app-works">
      <img className="phone-left" src={PhoneLeft} />
      <img className="phone-right" src={PhoneRight} />
    </div>
  );
};
```

```css
.phone-left {
  transform: translateY(-300px);
}

.phone-right {
  transform: translateY(-100px);
}
```

7. Aiz “How the app works” sekojošās 3 komponentes ir 1 komponente, kas noformēta atkarībā no padotā propa - attēls ir labajā vai kreisajā pusē. Loģika ir ļoti līdzīga **<Button />** komponentes, lai parādītu vienu vai otru pogas noformējumu.
Šajā gadījumā visērtāk būs izmantot **display: flex** on ar to saistītu īpašību **order**.

```jsx
import FeatureComponent from "../../components/FeatureComponent/FeatureComponent";
import FeatureOneImage from "../../assets/feature-one-image.png";
import FeatureTwoImage from "../../assets/feature-two-image.png";
import FeatureThreeImage from "../../assets/feature-three-image.png";

const Home = () => {
  return (
    <>
      <FeatureComponent
        featureImg={FeatureOneImage}
        preTitle="Create an account"
        title="Some text"
        text="Some text"
      />
      <FeatureComponent
        featureImg={FeatureTwoImage}
        preTitle="Explore varieties"
        title="Some text"
        text="Some text"
        imageIsRight
      />
      <FeatureComponent
        featureImg={FeatureThreeImage}
        preTitle="Checkout"
        title="Some text"
        text="Some text"
      />
    </>
  );
};
```

```jsx
const FeatureComponent = ({
  featureImg,
  imageIsRight,
  preTitle,
  title,
  text,
}) => {
  const componentClass = `feature-component ${
    imageIsRight ? "image-right" : ""
  }`;

  return (
    <div className={componentClass}>
      <img className="image" src={featureImg} />
      <div className="text">
        <p>{preTitle}</p>
        <p>{title}</p>
        <p>{text}</p>
      </div>
    </div>
  );
};
```

```css
.feature-component {
  display: flex;
}

.image {
  order: 1;
}

.text {
  order: 2;
}

.image-right .image {
  order: 2;
}

.image-right .text {
  order: 1;
}

@media only screen and (min-width: 768px) {
    .image {
        order: 1;
      }

      .text {
        order: 2;
      }

      .image-right .image {
        order: 1;
      }

      .image-right .text {
        order: 2;
      }
}
```

Par CSS **order** order īpašību ieteicams izlasīt šo īso rakstu:
https://css-tricks.com/almanac/properties/o/order/

8. Lai piešķirtu fona enojumu dāziem no attēliem, kas redzami dizainā, varm izmantot CSS box-shadow īpašību. Ir pieejami rīki, lai ģenerētu šai īpašībai nepieciešamās vērtības, piem.,
https://html-css-js.com/css/generator/box-shadow/

   Kaut arī šo īpašību nav sarežģīti izmantot, lai iepazītos ar box-shadow var izlasīt MDN dokumentāciju šai īpašībai:
https://developer.mozilla.org/en-US/docs/Web/CSS/box-shadow