Домашняя работа 2: Доступность в HTML

Студент: Даньшин Семён
Группа: K3440

Описание задания

Задание: улучшить доступность ранее реализованного сайта. Добавить необходимые HTML-атрибуты ко всему контенту на странице и проверить это с помощью инструментов из Dev Tools браузера Firefox и сервиса Google Lighthouse.

Теоретическая часть

Основные принципы доступности (a11y)

Доступность (accessibility, a11y) - это практика создания веб-приложений, которые могут использовать все люди, включая людей с ограниченными возможностями. Основные категории пользователей:

1. **Пользователи с нарушениями зрения** - используют программы экранного чтения (screen readers)
2. **Пользователи с нарушениями слуха** - нуждаются в субтитрах и визуальных альтернативах аудио
3. **Пользователи с моторными нарушениями** - используют клавиатуру вместо мыши
4. **Пользователи с когнитивными нарушениями** - нуждаются в простой и понятной навигации

Основные ARIA-атрибуты

- `aria-label` - текстовая метка для элемента
- `aria-labelledby` - ссылка на ID элемента, содержащего метку
- `aria-describedby` - ссылка на ID элемента с описанием
- `aria-hidden` - скрывает элемент от screen readers
- `role` - семантическая роль элемента
- `aria-live` - область с динамическим содержимым
- `aria-pressed` - состояние toggle-кнопки
- `aria-disabled` - состояние отключенного элемента
- `aria-expanded` - состояние раскрытого/свернутого элемента

Практическая часть: Примеры из проекта BankingThing

1. Доступность иконок

Иконки в проекте правильно помечены как декоративные элементы:

```
export const LogoutIcon = ({ size = 24, width, height, ...props }:
IconSvgProps) => (
  <svg
```

```
    aria-hidden="true"
    focusable="false"
    height={size || height}
    role="presentation"
    viewBox="0 0 24 24"
    width={size || width}
    fill="none"
    stroke="currentColor"
    strokeWidth="1.5"
    strokeLinecap="round"
    strokeLinejoin="round"
    {...props}
  >
    <path d="M15 3h4a2 2 0 0 1 2 2v14a2 2 0 0 1-2 2h-4" />
    <polyline points="10 17 15 12 10 7" />
    <line x1="15" y1="12" x2="3" y2="12" />
  </svg>
);
```

Примененные атрибуты:

- `aria-hidden="true"` - скрывает иконку от screen readers, так как она декоративная
- `focusable="false"` - предотвращает фокусировку на SVG элементе
- `role="presentation"` - указывает, что элемент носит исключительно презентационный характер

2. Доступность кнопок переключения темы

```
export const ThemeSwitch: FC<ThemeSwitchProps> = ({
  className,
  classNames,
}) => {
  const { theme, setTheme } = useTheme();
  const isSSR = useIsSSR();

  const onChange = () => {
    theme === "light" ? setTheme("dark") : setTheme("light");
  };

  const {
    Component,
    slots,
    isSelected,
    getBaseProps,
    getInputProps,
    getWrapperProps,
  } = useSwitch({
    isSelected: theme === "light" || isSSR,
    "aria-label": `Switch to ${theme === "light" || isSSR ? "dark" :
"light"} mode`,
    onChange,
```

```
  });

  return (
    <Component {...getBaseProps({...})}>
      <VisuallyHidden>
        <input {...getInputProps()} />
      </VisuallyHidden>
      <div {...getWrapperProps()}>
        {!isSelected || isSSR ? (
          <SunFilledIcon size={22} />
        ) : (
          <MoonFilledIcon size={22} />
        )}
      </div>
    </Component>
  );
};
```

Примененные атрибуты:

- `aria-label` - описывает действие переключателя для screen readers
- `VisuallyHidden` - компонент скрывает input визуально, но оставляет доступным для assistive technologies
- Динамическое изменение aria-label в зависимости от текущей темы

3. Доступность форм

В компоненте LoginCard используются семантические HTML-элементы и доступные поля ввода:

```
<Form className="flex flex-col gap-4" onSubmit={handleSubmit}>
  <Input
    isRequired
    label="Client ID"
    placeholder="Введите свой Client ID"
    type="text"
    value={clientId}
    onValueChange={setClientId}
    isDisabled={isLoading}
  />
  <Input
    isRequired
    label="Пароль"
    placeholder="Введите пароль"
    type="password"
    value={password}
    onValueChange={setPassword}
    isDisabled={isLoading}
  />
  {error && <p className="text-danger text-small">{error}</p>}
  <Button type="submit" color="primary" fullWidth isLoading={isLoading}>
    {isLoading ? "Вход..." : "Войти"}
```

```
  </Button>
</Form>
```

Применённые практики доступности:

- Использование семантического элемента `<Form>`
- `label` - каждое поле ввода имеет понятную метку
- `isRequired` - указывает обязательность поля
- `type="password"` - правильный тип для пароля
- `isDisabled` - атрибут отключения поля
- Отображение ошибок в доступном месте
- Состояния загрузки (`isLoading`) для обратной связи

4. Доступность интерактивных элементов

Кнопки в чате имеют правильные aria-атрибуты:

```
const renderActionControl = () => {
  if (shouldShowMic) {
    return (
      <Button
        isIconOnly
        radius="full"
        variant="flat"
        color={listening ? 'success' : 'default'}
        className="w-10 h-10"
        onPress={handleVoiceInput}
        isDisabled={!browserSupportsSpeechRecognition}
        aria-label={listening ? 'Остановить запись' : 'Начать голосовой
ввод'}
        title={browserSupportsSpeechRecognition ? undefined : 'Браузер не
поддерживает голосовой ввод'}
      >
        <MicIcon size={20} />
      </Button>
    );
  }

  return (
    <Button
      isIconOnly
      radius="full"
      color="primary"
      className="w-10 h-10"
      onPress={handleSendMessage}
      isDisabled={sendButtonState.disabled}
      isLoading={sendButtonState.status === 'loading'}
      data-status={sendButtonState.status}
      aria-label={sendButtonState.label}
      spinner={<Spinner classNames={{wrapper: "w-4 h-4"}} color="warning"
size="md" />}
```

```
    >
      <ArrowRightIcon size={20} />
    </Button>
  );
};
```

Примененные атрибуты:

- `aria-label` - динамическая метка, описывающая текущее состояние кнопки
- `title` - дополнительная подсказка при наведении
- `isDisabled` - отключение кнопки в зависимости от состояния
- Визуальная обратная связь через изменение цвета и состояния загрузки

5. Доступность навигации

Navbar использует семантическую разметку:

```
<HeroUINavbar maxWidth="xl" position="sticky">
  <NavbarContent className="basis-1/5 sm:basis-full" justify="start">
    <NavbarBrand as="li" className="gap-3 max-w-fit">
      <NextLink className="flex justify-start items-center gap-1"
href="/">
        <InteractiveLogo/>
      </NextLink>
    </NavbarBrand>
    <ul className="hidden sm:flex gap-4 justify-start ml-2">
      {pathname!="/auth" && siteConfig.navItems.filter(item => item.label
!== "Ассистент").map((item) => (
        <NavbarItem key={item.href}>
          <NextLink
            className={clsx(
              linkStyles({ color: "foreground" }),
              "data-[active=true]:text-primary data-[active=true]:font-
medium"
            )}
            color="foreground"
            href={item.href}
          >
            {item.label}
          </NextLink>
        </NavbarItem>
      ))}
    </ul>
  </NavbarContent>
  <NavbarContent>
    <Button className="bg-transparent hover:bg-transparent" isIconOnly
aria-label="Выйти" onPress={handleLogout}>
      <LogoutIcon className="text-default-500" />
    </Button>
  </NavbarContent>
</HeroUINavbar>
```

Применённые практики доступности:

- Использование семантических элементов `<nav>`, `<ul>`, `<li>`
- `aria-label` для кнопок с иконками
- Правильная структура навигации для screen readers
- Визуальное выделение активного пункта меню

6. Доступность нижней навигации

```
<Tabs
  aria-label="Bottom Navigation"
  items={siteConfig.navMenuItems}
  selectedKey={pathname}
  size="sm"
>
  {(item) => {
    const Icon = iconMap[item.icon];
    return (
      <Tab
        key={item.href}
        href={isAssistant ? undefined : item.href}
        onClick={isAssistant ? openFromBottom : undefined}
        title={
          <div className={`flex flex-col items-center ${assistCol}`}>
            {Icon && <Icon className={'w-6 h-6'} />}
            <span className={'text-[10px]'}>{item.label}</span>
          </div>
        }
      />
    );
  }}
</Tabs>
```

Примененные атрибуты:

- `aria-label="Bottom Navigation"` - описание назначения табов для screen readers
- Текстовые метки вместе с иконками для лучшего понимания
- Семантическое использование компонента Tabs

Проверка доступности

Инструменты проверки:

1. **Firefox DevTools Accessibility Inspector**
    - Проверка семантической структуры
    - Анализ контрастности цветов
    - Проверка порядка табуляции

2. **Google Lighthouse**
    - Автоматический аудит доступности
    - Оценка производительности и best practices
    - Рекомендации по улучшению
3. **Keyboard Navigation Test**
    - Проверка навигации по клавише Tab
    - Работа всех интерактивных элементов с клавиатуры
    - Видимые focus states

Основные метрики доступности:

- ✅ Все интерактивные элементы имеют aria-label
- ✅ Декоративные иконки скрыты от screen readers
- ✅ Формы используют семантические метки
- ✅ Кнопки имеют понятные описания состояний
- ✅ Навигация использует семантическую разметку
- ✅ Цветовой контраст соответствует WCAG 2.1 AA

Улучшения доступности в проекте

Реализованные улучшения:

1. **Семантическая разметка**
    - Использование правильных HTML-элементов (`nav`, `form`, `button`)
    - Логическая структура заголовков
2. **ARIA-атрибуты**
    - `aria-label` для всех иконок-кнопок
    - `aria-hidden` для декоративных элементов
    - Динамические aria-label для изменяющихся состояний
3. **Клавиатурная навигация**
    - Все элементы доступны через Tab
    - Правильный порядок фокусировки
    - Видимые focus states
4. **Обратная связь**
    - Состояния загрузки
    - Сообщения об ошибках
    - Визуальное подтверждение действий
5. **Responsive design**
    - Адаптивность для различных устройств
    - Доступность на мобильных устройствах

- Правильное масштабирование текста

Заключение

Проект BankingThing демонстрирует хорошие практики доступности:

1. **Структурная доступность** - использование семантических HTML-элементов и правильной иерархии
2. **Интерактивная доступность** - все элементы управления доступны с клавиатуры и имеют понятные метки
3. **Визуальная доступность** - достаточный контраст, понятная типографика, адаптивность
4. **Программная доступность** - правильное использование ARIA-атрибутов для assistive technologies

Доступность - это не дополнительная функция, а неотъемлемая часть качественного веб-приложения. Она делает продукт доступным для всех пользователей, независимо от их возможностей, и является важной частью пользовательского опыта.

Рекомендации для дальнейшего улучшения:

1. Добавить skip links для быстрого перехода к основному контенту
2. Реализовать живые области (aria-live) для динамического контента
3. Добавить альтернативный текст для всех изображений
4. Провести тестирование с реальными screen readers (NVDA, JAWS, VoiceOver)
5. Добавить больше контекстных подсказок для сложных интерфейсов