Домашняя работа 4: SVG-спрайт

Студент: Даньшин Семён
Группа: K3440
Дата: 12 января 2026 г.

Описание задания

Задание: вынести все используемые ранее SVG-иконки в общий SVG-спрайт. Если иконок не было, добавьте 3-5 иконок и поместите их в SVG-спрайт.

Теоретическая часть

Что такое SVG-спрайт?

SVG-спрайт - это один SVG-файл, содержащий множество иконок, каждая из которых определена как отдельный символ (`<symbol>`). Затем эти символы можно переиспользовать в разных местах документа через элемент `<use>`.

Преимущества SVG-спрайтов

1. **Производительность**
    - Один HTTP-запрос вместо множества
    - Уменьшение размера за счёт переиспользования общих путей
    - Браузер кеширует спрайт
2. **Масштабируемость**
    - Векторная графика масштабируется без потери качества
    - Адаптивность для Retina-дисплеев
3. **Удобство поддержки**
    - Все иконки в одном месте
    - Легко добавлять и удалять иконки
    - Централизованное управление стилями
4. **Гибкость стилизации**
    - Можно менять цвет через CSS (`fill`, `stroke`)
    - Можно применять CSS-анимации
    - Поддержка темизации через `currentColor`

Структура SVG-спрайта

```
<svg xmlns="http://www.w3.org/2000/svg" style="display: none;">
  <symbol id="icon-home" viewBox="0 0 24 24">
    <path d="..." />
```

```
  </symbol>

  <symbol id="icon-user" viewBox="0 0 24 24">
    <path d="..." />
  </symbol>

  <symbol id="icon-settings" viewBox="0 0 24 24">
    <path d="..." />
  </symbol>
</svg>
```

Использование SVG-спрайта

```
<!-- Встраивание спрайта в HTML -->
<div id="svg-sprite"></div>

<!-- Использование иконки -->
<svg width="24" height="24">
  <use href="#icon-home" />
</svg>
```

Практическая часть: Подход в проекте BankingThing

Текущий подход: React-компоненты

В проекте BankingThing используется альтернативный подход - каждая иконка реализована как отдельный React-компонент. Этот подход также имеет свои преимущества в контексте React-приложения.

Файл icons.tsx

```
import * as React from "react";
import { IconSvgProps } from "@/types";
```

Пример 1: Иконка выхода (LogoutIcon)

```
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>
);
```

Особенности:

- `aria-hidden="true"` - скрывает от screen readers
- `focusable="false"` - предотвращает фокус на SVG
- `role="presentation"` - указывает декоративную роль
- `currentColor` - цвет наследуется от родительского элемента

Пример 2: Логотип приложения (Logo)

```
export const Logo: React.FC<IconSvgProps> = ({
  size = 30,
  width,
  height,
  ...props
}) => (
  <svg
    height={size || height}
    viewBox="0 0 512 512"
    width={size || width}
    {...props}
  >
    <g>
      <path fill={props.fill == "colored"? "#17C964":undefined}
d="M441.596,46.582l12.666-29.813l-19.715-8.38L414.823,0l-12.675,29.814
        c-13.062,25.232-66.28-8.397-99.512,36.871c-30.024,40.913-
44.274,49.36-44.274,49.36s56.806,19.749,80.522-3.166
        c0,0-3.234,40.433,23.581,65.017c36.306,2.248,63.188-28.121,63.188-
28.121c2.308,38.387,53.571,60.182,53.571,60.182
        s-5.297-16.439,4.835-
66.128C494.805,91.2,431.296,70.794,441.596,46.582z"/>
      <path fill={props.fill == "colored"? "#7828C8":undefined}
d="M414.68,188.826c-17.357,9.845-35.759,14.436-53.723,13.324
        c-5.55-0.346-10.822-2.594-14.932-6.35c-15.782-14.494-23.75-32.668-
27.724-48.484
        c-19.421,2.737-31.658,2.914-43.895-0.11c-36.222,37.629-
112.894,72.689-149.976,94.005

C12.924,305.292,4.915,413.008,59.892,469.4c47.096,48.316,126.716,68.613,23
5.079-10.544
```

```
        c87.141-63.636,126.942-184.026,144.814-
241.69C429.696,210.639,424.416,203.118,414.68,188.826z M67.902,411.214
        c-12.876-117.881,115.27-139.761,115.27-
139.761S97.631,346.559,67.902,411.214z"/>
    </g>
  </svg>
);
```

Особенности:

- Условная окраска через props (`props.fill == "colored"`)
- Многоцветная иконка с несколькими путями
- Собственный viewBox для сложной графики

Пример 3: Иконки тем (MoonFilledIcon, SunFilledIcon)

```
export const MoonFilledIcon = ({
  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}
    {...props}
  >
    <path
      d="M21.53 15.93c-.16-.27-.61-.69-1.73-.49a8.46 8.46 0 01-1.88.13
8.409 8.409 0 01-5.91-2.82 8.068 8.068 0 01-1.44-8.66c.44-1.01.13-
1.54-.09-1.76s-.77-.55-1.83-.11a10.318 10.318 0 00-6.32 10.21 10.475
10.475 0 007.04 8.99 10 10 0 002.89.55c.16.01.32.02.48.02a10.5 10.5 0
008.47-4.27c.67-.93.49-1.519.32-1.79z"
      fill="currentColor"
    />
  </svg>
);

export const SunFilledIcon = ({
  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}
    {...props}
  >
    <g fill="currentColor">
      <path d="M19 12a7 7 0 11-7-7 7 7 0 017 7z" />
      <path d="M12 22.96a.969.969 0 01-1-.96v-.08a1 1 0 012 0 1.038 1.038
0 01-1 1.04zm7.14-2.82a1.024 1.024 0 01-.71-.29l-.13-.13a1 1 0 011.41-
1.41l.13.13a1 1 0 010 1.41.984.984 0 01-.7.29zm-14.28 0a1.024 1.024 0
01-.71-.29 1 1 0 010-1.41l.13-.13a1 1 0 011.41 1.41l-.13.13a1 1 0
01-.7.29zM22 13h-.08a1 1 0 010-2 1.038 1.038 0 011.04 1 .969.969 0 01-.96
1zM2.08 13H2a1 1 0 010-2 1.038 1.038 0 011.04 1 .969.969 0 01-.96
1zm16.93-7.01a1.024 1.024 0 01-.71-.29 1 1 0 010-1.41l.13-.13a1 1 0 011.41
1.41l-.13.13a.984.984 0 01-.7.29zm-14.02 0a1.024 1.024 0
01-.71-.29l-.13-.14a1 1 0 011.41-1.41l.13.13a1 1 0 010 1.41.97.97 0
01-.7.3zM12 3.04a.969.969 0 01-1-.96V2a1 1 0 012 0 1.038 1.038 0 01-1
1.04z" />
    </g>
  </svg>
);
```

Особенности:

- Использование `fill="currentColor"` для темизации
- Группировка путей через `<g>`
- Семантические названия компонентов

Пример 4: Интерактивные иконки

```
export const RefreshIcon: React.FC<IconSvgProps> = ({
  size = 24,
  width,
  height,
  ...props
}) => {
  return (
    <svg
      height={size || height}
      viewBox="0 0 24 24"
      width={size || width}
      fill="none"
      stroke="currentColor"
      strokeWidth="2"
      strokeLinecap="round"
      strokeLinejoin="round"
      {...props}
    >
      <path d="M3 12a9 9 0 0 1 9-9 9.75 9.75 0 0 1 6.74 2.74L21 8" />
      <path d="M21 3v5h-5" />
```

```
      <path d="M21 12a9 9 0 0 1-9 9 9.75 9.75 0 0 1-6.74-2.74L3 16" />
      <path d="M3 21v-5h5" />
    </svg>
  );
};

export const MicIcon: React.FC<IconSvgProps> = ({
  size = 24,
  width,
  height,
  ...props
}) => (
  <svg
    aria-hidden="true"
    focusable="false"
    height={size || height}
    role="presentation"
    viewBox="0 0 24 24"
    width={size || width}
    fill="none"
    {...props}
  >
    <path
      d="M12 3a3 3 0 0 1 3 3v5a3 3 0 0 1-6 0V6a3 3 0 0 1 3-3Z"
      stroke="currentColor"
      strokeWidth="2"
      strokeLinecap="round"
      strokeLinejoin="round"
    />
    <path
      d="M5 11v1a7 7 0 0 0 14 0v-1"
      stroke="currentColor"
      strokeWidth="2"
      strokeLinecap="round"
      strokeLinejoin="round"
    />
    <path
      d="M12 19v3"
      stroke="currentColor"
      strokeWidth="2"
      strokeLinecap="round"
      strokeLinejoin="round"
    />
  </svg>
);
```

Полный список иконок в проекте

Проект использует следующие иконки (всего 20+):

1. **Навигационные:**
    - LogoutIcon - выход из аккаунта

- ExpandLeftIcon - сворачивание панели
- ArrowRightIcon - отправка сообщения

2. **Действия:**
   - RefreshIcon - обновление данных
   - TrashIcon - удаление
   - PlusIcon - добавление
   - PlayIcon - запуск
3. **Статусы:**
   - CheckCircleIcon - успешное выполнение
   - ExclamationTriangleIcon - предупреждение
   - ClockIcon - время/ожидание
4. **Интерфейс:**
   - SearchIcon - поиск
   - MicIcon - голосовой ввод
   - SunFilledIcon - светлая тема
   - MoonFilledIcon - темная тема
5. **Социальные:**
   - GithubIcon - ссылка на GitHub
   - TwitterIcon - ссылка на Twitter
   - DiscordIcon - ссылка на Discord
6. **Бренд:**
   - Logo - логотип приложения
   - HeartFilledIcon - избранное

Использование иконок в компонентах

Пример использования в navbar.tsx

```
import { LogoutIcon } from "@/components/icons";

<Button
  className="bg-transparent hover:bg-transparent"
  isIconOnly
  aria-label="Выйти"
  onPress={handleLogout}
>
  <LogoutIcon className="text-default-500" />
</Button>
```

Пример использования в ThemeSwitch

```
import { SunFilledIcon, MoonFilledIcon } from "@/components/icons";

<div {...getWrapperProps()}>
  {!isSelected || isSSR ? (
    <SunFilledIcon size={22} />
  ) : (
    <MoonFilledIcon size={22} />
  )}
</div>
```

Пример использования в InteractiveChat

```
import { MicIcon, ArrowRightIcon } from "@/components/icons";

<Button
  isIconOnly
  onPress={handleVoiceInput}
  aria-label={listening ? 'Остановить запись' : 'Начать голосовой ввод'}
>
  <MicIcon size={20} />
</Button>
```

Сравнение подходов: SVG-спрайт vs React-компоненты

SVG-спрайт

Преимущества:

- ✅ Один HTTP-запрос
- ✅ Меньший общий размер файла
- ✅ Браузерное кеширование
- ✅ Работает без JavaScript

Недостатки:

- ❌ Сложность в TypeScript/React
- ❌ Нет type safety
- ❌ Сложнее управлять props
- ❌ Требует дополнительной настройки

React-компоненты (используется в проекте)

Преимущества:

- ✅ Type safety через TypeScript
- ✅ Удобство использования в JSX
- ✅ Легко передавать props (size, color, className)

- ✅ Tree-shaking (неиспользуемые иконки не попадают в bundle)
- ✅ Автокомплит в IDE
- ✅ Модульность и переиспользуемость

Недостатки:

- ❌ Больше кода в bundle (если не используется tree-shaking)
- ❌ Множество компонентов вместо одного файла

Альтернативный подход: Создание SVG-спрайта для проекта

Если бы мы решили использовать SVG-спрайт, вот как это могло бы выглядеть:

sprite.svg

```
<svg xmlns="http://www.w3.org/2000/svg" style="display: none;">
  <symbol id="icon-logout" viewBox="0 0 24 24">
    <path d="M15 3h4a2 2 0 0 1 2 2v14a2 2 0 0 1-2 2h-4"
          stroke="currentColor"
          fill="none"
          stroke-width="1.5"/>
    <polyline points="10 17 15 12 10 7"
              stroke="currentColor"
              fill="none"
              stroke-width="1.5"/>
    <line x1="15" y1="12" x2="3" y2="12"
          stroke="currentColor"
          stroke-width="1.5"/>
  </symbol>

  <symbol id="icon-sun" viewBox="0 0 24 24">
    <g fill="currentColor">
      <path d="M19 12a7 7 0 11-7-7 7 7 0 017 7z" />
      <path d="M12 22.96a.969.969 0 01-1-.96v-.08a1 1 0 012 0 1.038 1.038
0 01-1 1.04z"/>
    </g>
  </symbol>

  <symbol id="icon-moon" viewBox="0 0 24 24">
    <path d="M21.53 15.93c-.16-.27-.61-.69-1.73-.49..."
fill="currentColor" />
  </symbol>

  <symbol id="icon-mic" viewBox="0 0 24 24">
    <path d="M12 3a3 3 0 0 1 3 3v5a3 3 0 0 1-6 0V6a3 3 0 0 1 3-3Z"
          stroke="currentColor"
          fill="none"/>
  </symbol>

  <symbol id="icon-refresh" viewBox="0 0 24 24">
    <path d="M3 12a9 9 0 0 1 9-9 9.75 9.75 0 0 1 6.74 2.74L21 8"
          stroke="currentColor"
```

```
            fill="none"/>
    </symbol>
  </svg>
```

Icon.tsx компонент для использования спрайта

```
interface IconProps {
  name: string;
  size?: number;
  className?: string;
}

export const Icon: React.FC<IconProps> = ({ name, size = 24, className })
=> {
  return (
    <svg
      width={size}
      height={size}
      className={className}
      aria-hidden="true"
    >
      <use href={`#icon-${name}`} />
    </svg>
  );
};

// Использование
<Icon name="logout" size={24} className="text-default-500" />
```

Заключение

Выбранный подход в проекте BankingThing

Проект использует подход с React-компонентами, что является оптимальным решением для современного React/Next.js приложения по следующим причинам:

1. **Type Safety** - TypeScript проверяет правильность использования компонентов
2. **Developer Experience** - автокомплит, подсказки типов, легкость рефакторинга
3. **Модульность** - каждая иконка - независимый модуль
4. **Tree Shaking** - неиспользуемые иконки не попадают в финальный bundle
5. **Гибкость** - легко передавать props и изменять поведение

Когда использовать SVG-спрайты

SVG-спрайты лучше подходят для:

- Статических сайтов без фреймворков
- Большого количества одинаковых иконок (100+)
- Критичной оптимизации начальной загрузки

- Старых браузеров без поддержки современного JavaScript

Рекомендации

Для React-приложений рекомендуется:

1. Использовать React-компоненты для иконок (как в проекте)
2. Применять TypeScript для type safety
3. Использовать `currentColor` для темизации
4. Добавлять accessibility-атрибуты (`aria-hidden`, `role="presentation"`)
5. Оптимизировать SVG-код (удалять лишние атрибуты, упрощать пути)

Проект BankingThing демонстрирует современный и эффективный подход к работе с SVG-иконками в React-экосистеме.