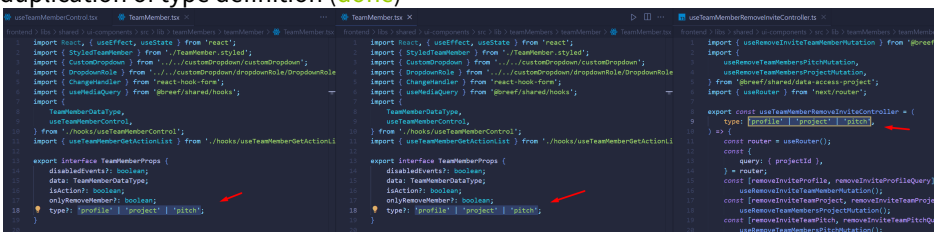
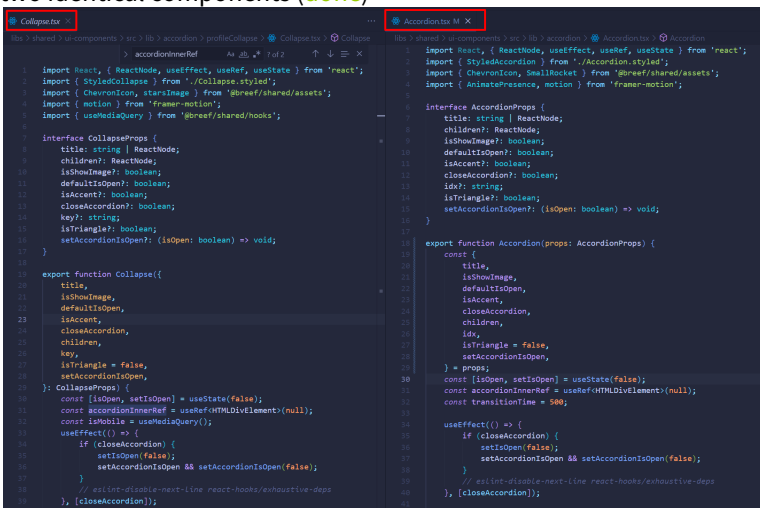
2023-05-23 Frontend [WIP]

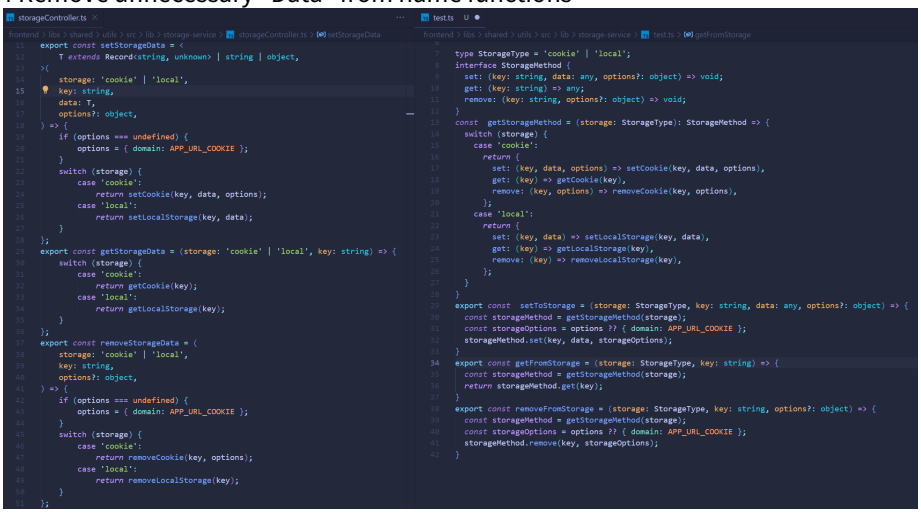
Breef (Curated)

Exported on 07/04/2023

Table of Contents

1. Date	23 мая 2023 г.	Notes
2. Repo	**frontend**	https://gitlab.light-it.tools/breef/frontend
Repository setup(Gitflow)	**OK**	
Sonar	**OK**	
Following the defined architecture	**HAS ISSUES**	
Churn	-	
Cognitive complexity	**OK**	
Cyclomatic complexity	**OK**	

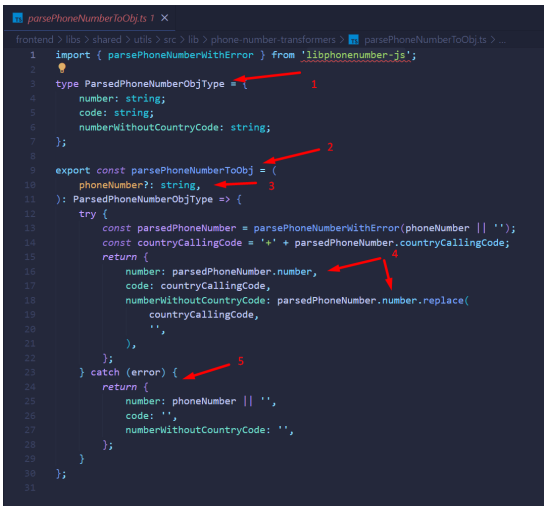
Code duplication (DRY)	**OK**	• We can incapsulate duplicated code in separate place (done)  • duplication of type definition (done)  • two identical components (done)  • duplication (done) 
KISS	**OK**	

Following standards	**HAS ISSUES**	• We can improve this solution (original on the left side) (done) 1 Introduced the StorageType type alias to represent the storage type ('cookie' or 'local'). 2 Introduced the StorageMethod interface to represent the shape of storage methods 3 Created the getStorageMethod function that returns the appropriate storage method object based on the storage type 4 Remove unnecessary "Data" from name functions  • we can eliminate the redundant `return false` statement and we can use key property instead of keyCode property of event object. Check import please. (done) 

- we can remove unnecessary if/else statement and redundant return. Also I propose to change the name of the function for example it can be `FilterLinksByType` (done)

- we need to simplify this function and then it will be easier for understanding and more readable also we need to change the name to convertAdvantages
- (done)
-

- 1 we need to change type name for my opinion it'll be great to use Phone or PhoneNumber
  2 change name of the function parsePhoneNumber or processPhoneNumber
  3 optional parameter is not correct
  4 we can combine it in one identificator and reuse it
  5 Why are we not interested in error?
- (done)

- 

- I propose to change function name, don't use unnecessary "?" operator, simplify conditions
- (done)
- 

- 1 change name of the function 2 change name of ValueArray 3 remove "?" 4 I believe we don't need to use second if instead of it we need to use else
- (done)

-

formatText.ts 1

libs > shared > utils > src > lib > format-text > formatText.ts > shotBudgetToMinMax

```
};

export const shotBudgetToMinMax = (value: string) => {
    if (!value) return { min: 0, max: 0 };
    const valueArray = value?.split('-');
    let maxBudget = 0;
    let minBudget = 0;

    if (valueArray?.length < 2) {
        minBudget = numeral(valueArray[0].trim().replace('+', '')).value() || 0;
        maxBudget = minBudget && minBudget * 1.2;
    }
    if (valueArray?.length >= 2) {
        minBudget = numeral(valueArray[0].trim()).value() || 0;
        maxBudget = numeral(valueArray[1].trim()).value() || 0;
    }

    return {
        min: minBudget,
        max: maxBudget,
    };
};
```

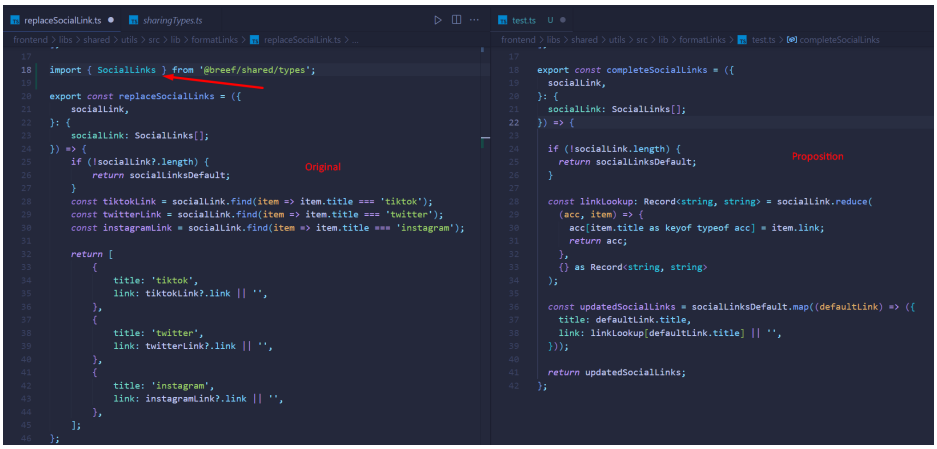
- It will be great to use enum or typing (now we use "magic string")
- (done)
-

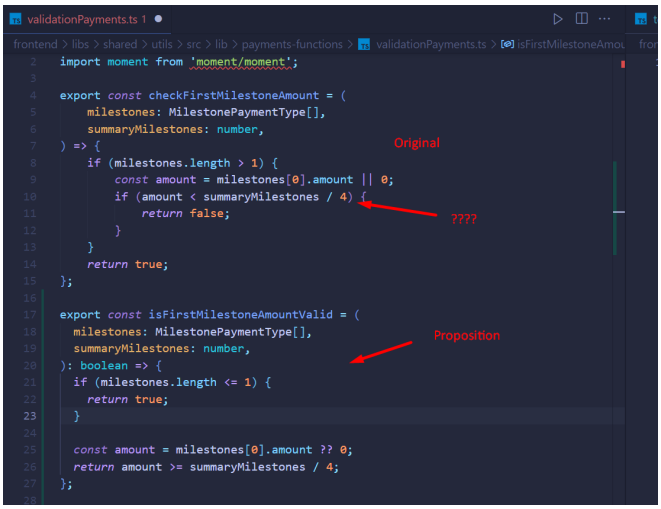
formatBrandDocumentsIcon.tsx 1

frontend > libs > shared > utils > src > lib > formatBrandDocuments > formatBrandDocumentsIcon.tsx > ...

```
import { FileIcon, PdfIcon } from '@breef/shared/assets';

export const getBrandDocumentIcon = (linkTitle: string) => {
    const linkType = linkTitle.split('.');

    if (linkType[linkType.length - 1] === 'pdf') {
        return <PdfIcon className="pdf-icon" data-testid="pdf-icon" />;
    }
    if (linkType[linkType.length - 1] === 'doc') {
        return <FileIcon className="pdf-icon" data-testid="file-icon" />;
    }
    if (linkType[linkType.length - 1] === 'docx') {
        return <FileIcon className="pdf-icon" data-testid="file-icon" />;
    }
    return null;
};
```

- We need to change name SocialLinks to SocialLink also take into account one of possible solution of refactoring. It is necessary to change the name of function
- (done)

- 

- It'll be better to change name function name. What is 4 (magic number. We need to use undersandable identificator for it) ?
- (done)
- 

- add typings and make function without inner conditions. We need to change the name of the function
- (done)

- validationPayments.ts 3

frontend > libs > shared > utils > src > lib > payments-functions > validationPayments.ts > ...

```ts
export const checkFirstMilestoneMonth = (
    milestones: MilestonePaymentType[],
) => {
    if (milestones.length > 1) {
        const milestonesDates = milestones.map(item =>
            moment(item.invoiceDate),
        );
        const minDate = moment.min(milestonesDates);
        if (milestonesDates[0] > minDate) {
            return false;
        }
    }
    return true;
};
```

Original

```ts
export const checkFirstMilestoneMonth = (milestones: MilestonePaymentType[]): boolean => {
  if (milestones.length <= 1) {
    return true;
  }

  const milestonesDates = milestones.map((item) => moment(item.invoiceDate));
  const minDate = moment.min(milestonesDates);

  return milestonesDates[0] <= minDate;
};
```

Proposition

- Lets remove unnecessary code and make it simlier
- (done)
- calculatePayments.ts 2

frontend > libs > shared > utils > src > lib > payments-functions > calculatePayments.ts > calculateSummaryMilestones

```ts
import { MilestonePaymentType, RetainerPaymentType } from '@breef/shared/types';

export const calculateSummaryMilestones = (
    milestones: MilestonePaymentType[],
) => {
    if (milestones.length !== 0) {
        return milestones
            .map(item =>
                !isNaN(Number(item.amount)) ? Number(item.amount) : 0,
            )
            .reduce((acc, curr) => acc + curr);
    }
    return 0;
};
```

original

```ts
export const calculateSummaryMilestones = (milestones: MilestonePaymentType[]): number => {
  return milestones.reduce((acc, curr) => acc + (Number(curr.amount) || 0), 0);
};
```

Proposition

- We need to make this code simple
- (done)

-

```
};

export const calculateSummaryRetainer = (
    retainer: RetainerPaymentType | null,
) => {                                                    Original
    const amount = Number(retainer?.amount);
    const numberOfPayments = Number(retainer?.numberOfPayments);
    if (retainer && !isNaN(amount)) {
        if (retainer.numberOfPayments !== '0' && !isNaN(numberOfPayments)) {
            return amount * numberOfPayments;
        }
        return amount;
    }
    return 0;
};

export const calculateSummaryRetainer = (retainer: RetainerPaymentType | null): number => {
    if (!retainer) {
        return 0;
    }                                                     Proposition

    const amount = Number(retainer.amount);
    const numberOfPayments = Number(retainer.numberOfPayments);

    if (isNaN(amount)) {
        return 0;
    }

    if (numberOfPayments !== 0 && isNaN(numberOfPayments)) {
        return amount;
    }

    return amount * numberOfPayments;
};
```

- We can use this changes in useEffect

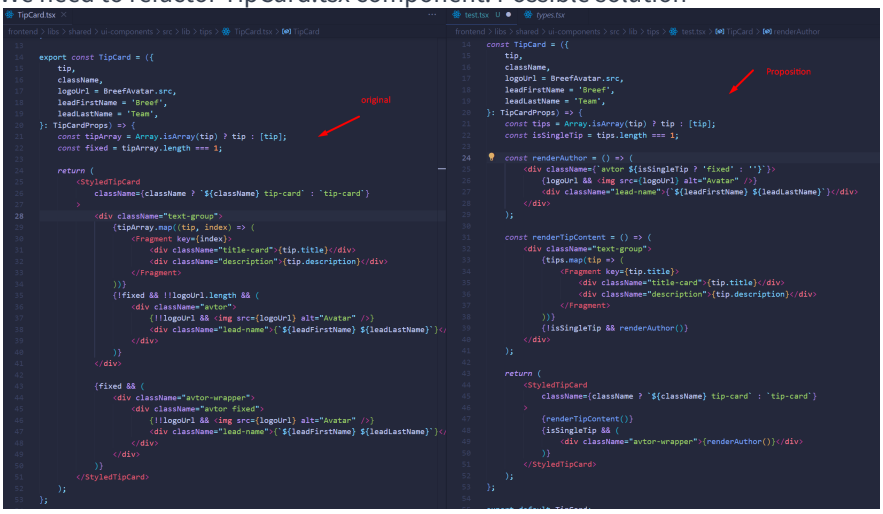
```
Tooltip.tsx
frontend > libs > shared > ui-components > src > lib > tooltip > Tooltip.tsx > Tooltip

    const styleError = {
        width: 'max-content',
        backgroundColor: `${colors.mainError}`,
        border: 'none',
        color: `${colors.mainWhite}`,
    };

    useEffect(() => {                                     Original
        if (isError) return setOpen(true);
        return setOpen(false);
    }, [isError]);

    useEffect(() => {                                     Proposition
        setOpen(isError || false);
    }, [isError]);

    const { x, y, reference, floating, context } = useFloating({
        placement,
        open,
        onOpenChange: !isError ? setOpen : undefined,
```

- We don't need to use !! operator and also we need to check for noteIcon is added to prevent rendering the span element when noreicon is falsy

```
TitleStep.tsx
frontend > libs > shared > ui-components > src > lib > title > TitleStep.tsx > TitleStep

    className,
    noteIcon,
    subTitle,
}: TitleStepProps) => {
    return (
        <StyledTitleStep className={className}>
            {!!step && !!numberSteps && (
                <div className="step-number">
                    {step}/{numberSteps}
                </div>
            )}
            {step && numberSteps && (                     Proposition
                <div className="step-number">
                    {step}/{numberSteps}
                </div>
            )}
            <div className="step-title-wrapper">
                <h2 className="step-title">
                    {title}
                    <span className="note-icon">{noteIcon}</span>
                    {noteIcon && <span className="note-icon">{noteIcon}</span>}
                </h2>
                {subTitle && <h4 className="step-subtitle">{subTitle}</h4>}
            </div>
        </StyledTitleStep>
    );
};

export default TitleStep;
```

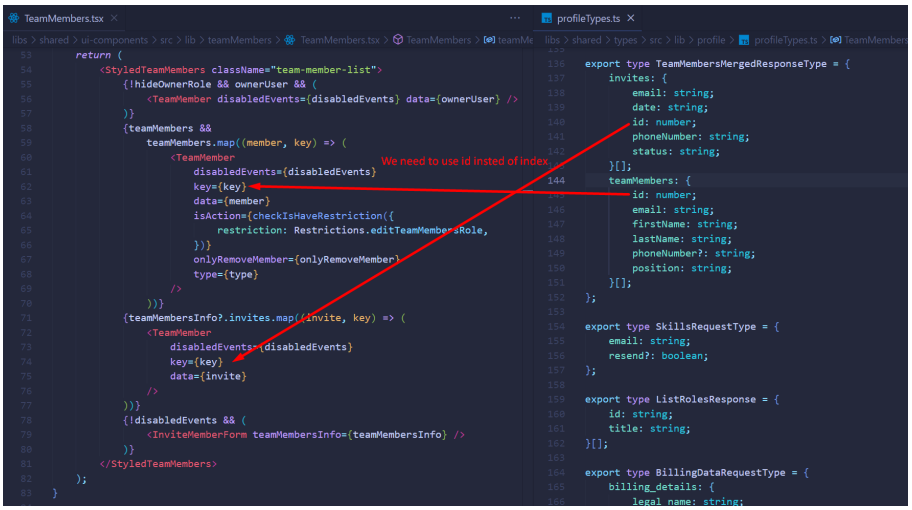
- We can use solution without switch operator

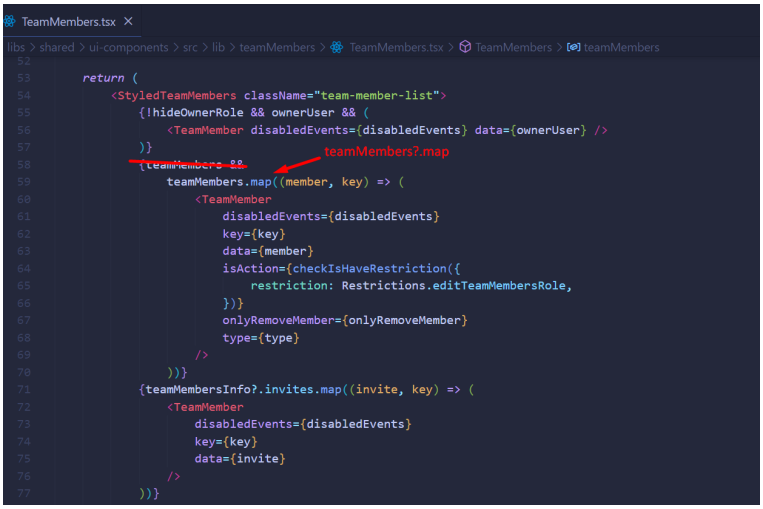
- We need to refactor TipCard.tsx component. Possible solution

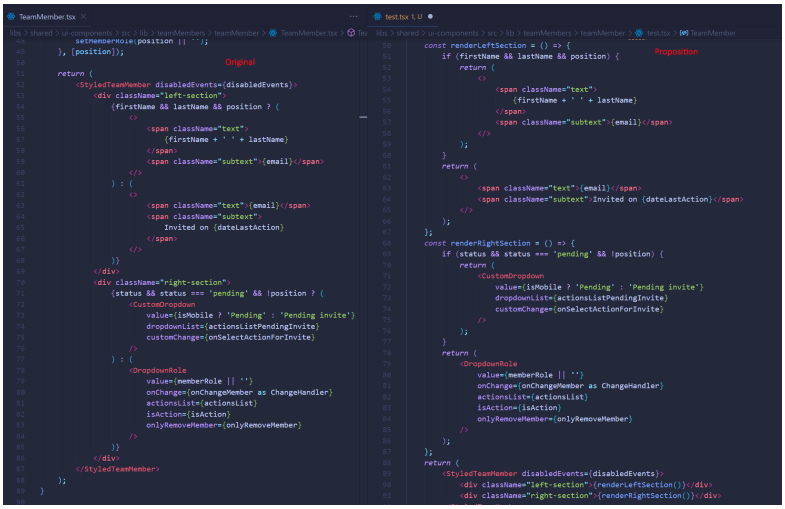


- We need to avoid using on unnecessary else and make code simpler (done)

- TeamMember.tsx (done)







- TabsNavigation.tsx
  1 this will make our code simpler and we will not use an extra else (done)

2 this proposition make our solution more readable and we don't use inner if (done)

3 Why we use config here and we get it as props of component (done)

TabsNavigation.tsx

```
libs > shared > ui-components > src > lib > tabsNavigation > TabsNavigation.tsx > TabsNavigation
                    undefined,
                    { shallow: true },
                )
                .then(() => null);
        },
        // eslint-disable-next-line react-hooks/exhaustive-deps
        [router.pathname],
    );

    useEffect(() => {
        if (router.isReady) {
            if (
                !query.tab ||
                !config.some(item => item.tab === query.tab) ||
                config.find(item => item.tab === query.tab)?.disabled
            ) {
                onChangeTab(config[0].tab);
            }
        }
    }, [router.isReady, config, onChangeTab, query.tab]);
```

4 in this case won't render unnecessary code and also we need to change (done)

- in this case we'll get better clarity and readability (done)

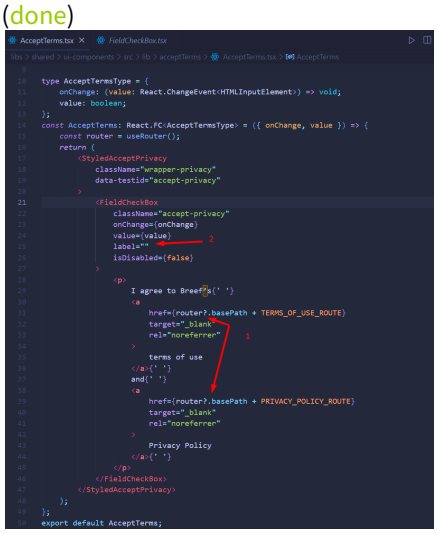
AccentNumber.tsx 2

```
shared > ui-components > src > lib > accentNumber > AccentNumber.tsx > AccentNumber > transfo
import { StyledAccentNumber } from './AccentNumber.styled';

export function AccentNumber({
    number,
    isOptional,
}: {
    number: number;
    isOptional?: boolean;
}) {
    const transformToAccentFormat = (number: number) => {
        const stringValue = number + '';
        return '0'.repeat(3 - stringValue.length) + stringValue;
    };

    const transformToAccentFormat = (number:number) => {
        const value = String(number);
        return '0'.repeat(3 - value.length) + value;
    };
```

Original

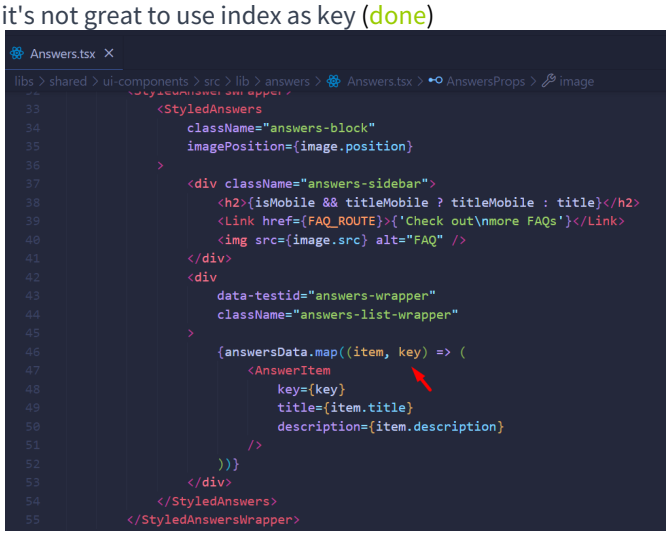
Proposition

- 1 think in this case router is not optional 2 we don't need to pass empty string (done)



- We need to change names in interface. Then our code will be descriptive (done)

- it's not great to use index as key (done)



- I think in current case it'll be better to use one of next solutions (screen 2, 3). In this case we'll get stronger typings (done)

```tsx
const clientActionButtonCtaStatuses: ActionButtonConfigType = {
    activeProjects: {
        text: 'View Projects',
        description: `YOU HAVE ${projectsCount} ACTIVE PROJECTS`,
        onClick: () => scrollToProjectSettingsBar(),
        imageConfig: {
            imageUrl: ctaGlobeImage.src,
            position: {
                top: 0,
                right: 20,
            },
        },
    },
    projectPlaning: {
        text: 'Grab Time',
        description: 'BOOK A PLANNING CALL',
        onClick: () => toggleBookACallPopup(),
        imageConfig: {
```

```tsx
export type ActionButtonConfigType = {
    [key: string]: {
        text: string | ReactNode;
        description?: string;
        descriptionMore?: string;
        onClick?: (meta: MetadataType) => void;
        imageConfig?: {
            imageUrl: string;
            position: {
                top?: number;
                bottom?: number;
                right?: number;
            };
        };
    };
};
```

```tsx
export type ActionButtonNameType =
    | 'activeProjects'
    | 'projectPlaning'
    | 'startProject';

export type ActionButtonConfigType1 = {
    text: string | ReactNode;
    description?: string;
    descriptionMore?: string;
    onClick?: (meta: MetadataType) => void;
    imageConfig?: {
        imageUrl: string;
        position: {
            top?: number;
            bottom?: number;
            right?: number;
        };
    };
};

export type test = {
    [k in ActionButtonNameType]?: ActionButtonConfigType1;
};
```

Proposition 1

```tsx
// or
export type clientActionButtonCtaStatuses =
    | 'activeProjects'
    | 'projectPlaning'
    | 'startProject';

export type agencyActionButtonCtaStatuses = 'activeProjects' | 'signUp' | 'incompleteProfile';

export type ActionButtonConfigType2 = {
    text: string | ReactNode;
    description?: string;
    descriptionMore?: string;
    onClick?: (meta: MetadataType) => void;
    imageConfig?: {
        imageUrl: string;
        position: {
            top?: number;
            bottom?: number;
            right?: number;
        };
    };
};

type PartialRecord<K extends keyof any, T> = {
  [P in K]? : T
}

type clientActionButtonCtaStatusesType = PartialRecord<clientActionButtonCtaStatuses, ActionButtonConfigType2>
type agencyActionButtonCtaStatusesType = PartialRecord<agencyActionButtonCtaStatuses, ActionButtonConfigType2>
type client  = Record<clientActionButtonCtaStatuses, ActionButtonConfigType1>
```

Proposition 2

- The same problem (done)

useActionButtonProjectConfig.tsx

```
libs > shared > hooks > src > lib > useActionButtonProjectConfig.tsx > useActionButtonProjectConfig > clientProjectA
41      const { changePage } = useRouteControl();
42      const clientProjectActionButtonStatuses: ActionButtonConfigType = {
43          continuePost: {
44              text: 'Review + Post',
45              description: 'POST YOUR PROJECT',
46              onClick: meta => {
47                  resetProjectEdit();
48                  changePage(
49                      PROJECT_EDIT_ROUTE.reverse({
50                          projectId: meta.projectId || 0,
51                      }) || '',
52                  );
53              },
54              imageConfig: {
55                  imageUrl: ctaManOrangeImage.src,
56                  position: {
57                      right: 0,
58                      bottom: 12,
59                  },
60              },
61          },
```

- I propose to simplify the current solution and also change the variable names to improve the readability of the code. (done)

useChecksContainsUrlPath.tsx 2 — Original

```
import { useRouter } from 'next/router';
import React, { useEffect, useState } from 'react';

export const useCheckIsContainsUrlPath = ({
    pathNames,
}: {
    pathNames: string[];
}) => {
    const router = useRouter();
    const [isContainsPathName, setIsContainsPathName] = useState(false);

    const checkIsContainsUrlPath = () => {
        const arrLength = pathNames.length - 1;
        for (let i = 0; i <= arrLength; i++) {
            if (router.pathname.includes(pathNames[i])) {
                setIsContainsPathName(true);
                return;
            }
            setIsContainsPathName(false);
        }
    };

    useEffect(() => {
        if (pathNames.length) {
            checkIsContainsUrlPath();
        }
    }, [pathNames]);

    return { isContainsPathName };
};
```

test.tsx 1, U — Proposition

```
import { useRouter } from 'next/router';
import { useEffect, useState } from 'react';

export const useUrlPathContains = ({ pathNames }: { pathNames: string[] }) => {
    const router = useRouter();
    const [hasMatchedPath, setHasMatchedPath] = useState(false);

    const checkIsContainsUrlPath = () => {
        const containsPath = pathNames.some(pathName =>
            router.pathname.includes(pathName),
        );
        setHasMatchedPath(containsPath);
    };

    useEffect(() => {
        if (pathNames.length) {
            checkIsContainsUrlPath();
        }
    }, [pathNames]);

    return { hasMatchedPath };
};
```

- NEW
- Добавил интерфейс, изменил название переменной

AccentNumber.tsx

```
import { StyledAccentNumber } from './AccentNumber.styled';

export function AccentNumber({
    number,
    isOptional,
}: {
    number: number;
    isOptional?: boolean;
}) {
    const transformToAccentFormat = (number: number) => {
        const stringValue = number + '';
        return '0'.repeat(3 - stringValue.length) + stringValue;
    };
    return (
        <StyledAccentNumber className="accent-number">
            <span className="round" />
            <span className="number">
                {transformToAccentFormat(number)}
                {isOptional && ' (optional)'}
            </span>
        </StyledAccentNumber>
    );
}
```

test.tsx

```
import { StyledAccentNumber } from './AccentNumber.styled';

function transformToAccentFormat(number: number): string {
    const formattedNumber = number.toString();
    return '0'.repeat(3 - formattedNumber.length) + formattedNumber;
}

interface AccentNumberProps {
    number: number;
    isOptional?: boolean;
}

export function AccentNumber({ number, isOptional }: AccentNumberProps) {
    return (
        <StyledAccentNumber className="accent-number">
            <span className="round" />
            <span className="number">
                {transformToAccentFormat(number)}
                {isOptional && ' (optional)'}
            </span>
        </StyledAccentNumber>
    );
}
```

- Возможно pageTitle будет лучше именем

Head.tsx 1

libs > shared > ui-components > src > lib > animateLayoutPage > head > Head.tsx > HeadApp

```tsx
import Head from 'next/head';

export default function HeadApp({ title }: { title: string }) {
    const titleContent = `breef. | ${title}`;

    const pageTitle = `breef. | ${title}`;
    return (
        <Head>
            <title>{titleContent}</title>
            <meta
                key="viewport"
                name="viewport"
                content="width=device-width, initial-scale=1, shrink-to-fit=no"
            />
            <link rel="icon" type="image/x-icon" href="/favicon.ico" />
            <link rel="preload" as="font" />
        </Head>
    );
}
```

- Поменять имя интерфейса (соответствует рекомендациям React) добавить функцию обработки события

AnswerItem.tsx 2

libs > shared > ui-components > src > lib > answers > answerItem > AnswerItem.tsx > AnswerItem

```tsx
interface AnswersProps {
    title: string;
    description: string;
}

interface AnswerItemProps {
    title: string;
    description: string;
}

export function AnswerItem({ title, description }: AnswersProps) {
    const [isOpen, setIsOpen] = useState(false);

    const toggleOpen = () => {
        setIsOpen(!isOpen);
    };

    return (
        <StyledAnswerItem
            data-testid="answer-item"
            onClick={() => setIsOpen(!isOpen)}
            isOpen={isOpen}
        >
            <div className="answer-header">
                <h2>{title}</h2>
                <ChevronSmallStraight />
```

1

2

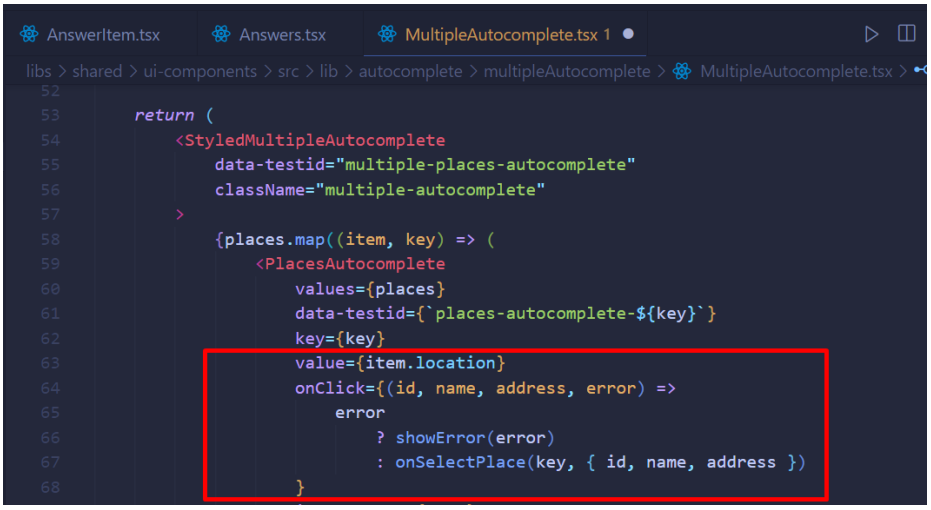
- вынес обработку заголовка в отдельную переменную, не использовать index(key) в качестве key property

AnswerItem.tsx | Answers.tsx 1 | test.tsx

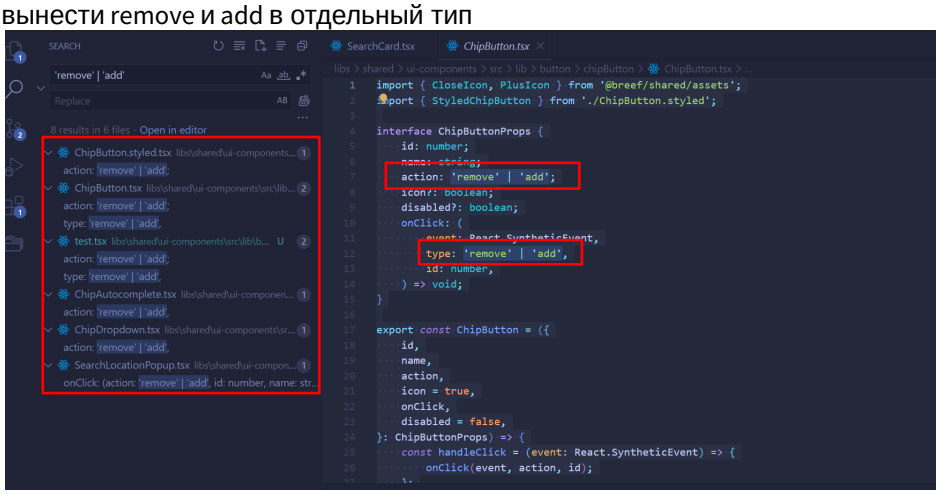
libs > shared > ui-components > src > lib > answers > Answers.tsx > Answers

```tsx
}: AnswersProps) {
    const { isMobile } = useContext(MediaContext);

    const renderTitle = isMobile && titleMobile ? titleMobile : title;

    return (
        <StyledAnswersWrapper>
            <StyledAnswers
                className="answers-block"
                imagePosition={image.position}
            >
                <div className="answers-sidebar">
                    <h2>{isMobile && titleMobile ? titleMobile : title}</h2>
                    <Link href={FAQ_ROUTE}>{'Check out\nmore FAQs'}</Link>
                    <img src={image.src} alt="FAQ" />
                </div>
                <div
                    data-testid="answers-wrapper"
                    className="answers-list-wrapper"
                >
                    {answersData.map((item, key) => (
                        <AnswerItem
                            key={key}
                            title={item.title}
                            description={item.description}
                        />
                    ))}
```

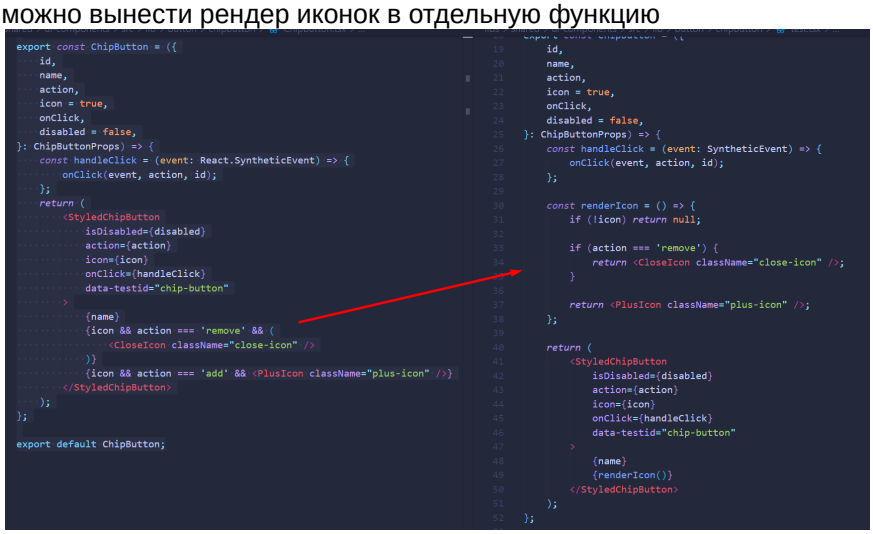
- можно добавить новый интерфейс, стоит вынести в отдельный обработчик клика



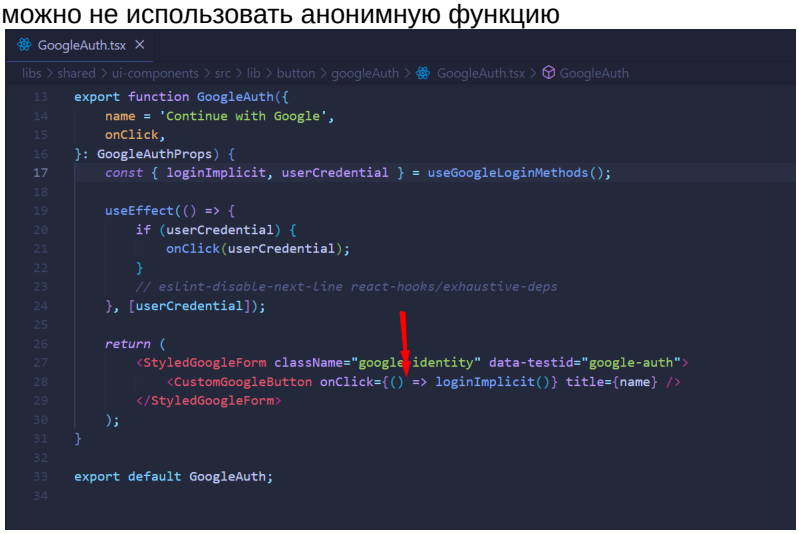
- вынести remove и add в отдельный тип



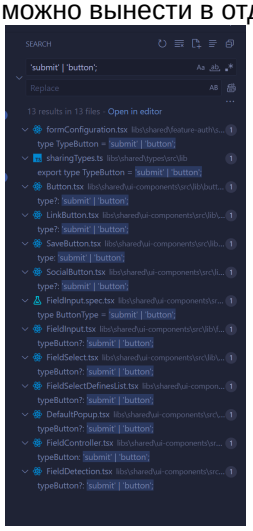
- можно вынести рендер иконок в отдельную функцию



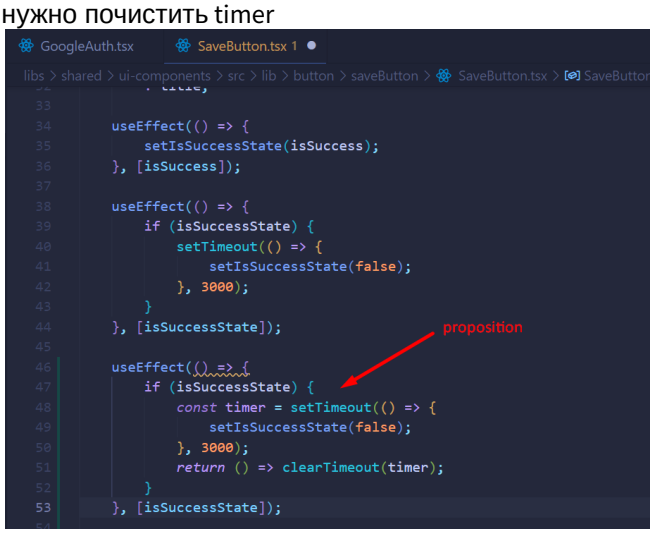
- можно не использовать анонимную функцию

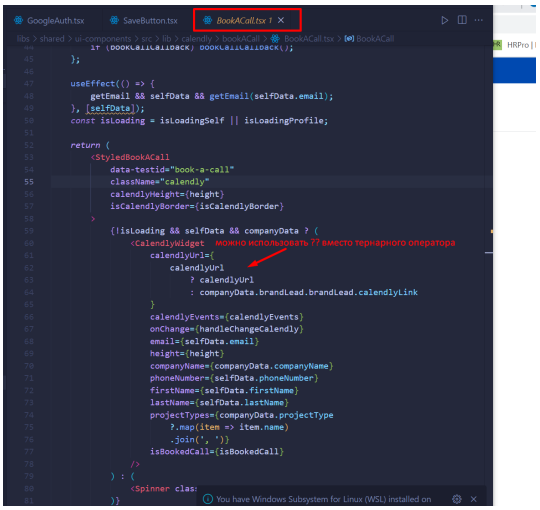


- можно вынести в отдельный тип



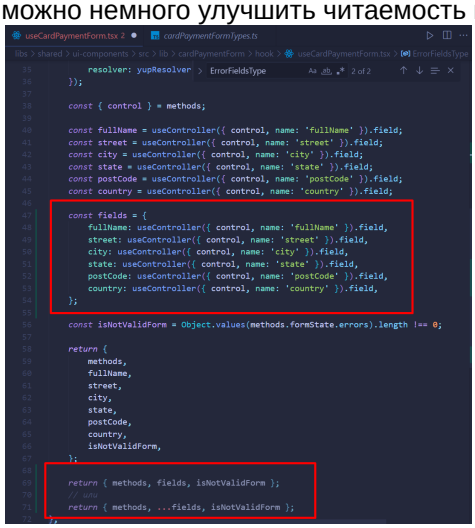
- нужно почистить timer





- странное решение

- можно немного улучшить читаемость кода,



- можно ли использовать вот такой интерфейс или нет

Clean (code-level)	**OK**	
Security (code-level)	**OK**	
Tests	**FAIL**	
Documentation (code-level)	**HAS ISSUES**	
Author	[1]Vakula Pavel	

1 https://confluence.light-it.tools/display/~vakulapavel0