10. Default опасность

**Export default <name>** может создать путаницу, т.к. при импорте <name> может называться произвольно. Очень важно правильно указать путь, откуда мы импортируем!

При импорте

**import <name> from ‘./folder/file.ts’** важно правильно указать имя и путь!

Импорт компонент можно делать автоматически:

**ALT + Enter** - автоимпорт компонент в WebStorm. При автоимпорте **проверяй path!!!**

11. Соц сеть, шаг 1, верстка, grid

**Делаем верстку на любой технологии, можно на FLEX, GRID, FLOAT.**

**В дано случае будем делать верстку на GRID**

12. Разбивка на компоненты.

Теперь разбиваем нашу разметку на компоненты.

Для этого создадим отдельную папку, в ней отдельные файлы, название пишем **с большой буквы!**

**Не забываем делать импорт-экспорт.**

13. Git

**Ссылка на репозиторий Димыча (там есть TYPESCRIPT):**

https://github.com/it-kamasutra/react-way-of-samurai

**Модули node** не закидываются в репозиторий, автоматом прописывается в GITIGNORE! Но в файл package.json сохраняются все зависимости! Из него node Подтягивает инфо какие модули установить.

Пишем NPM install для установки зависимостей, т.к. при создании проекта их нет

14. CSS Modules

Лучше создавать компоненты, и к ним писать отдельные CSS стили.

В файле компоненты нуно указывать импортируемый стиль CSS, например:

import './Navbar.css'**;**

Но, даже без импорта при совпадении стилей, они будут влиять на классы с таким же названием. Поэтому, если нужно, чтобы CSS стили были прописаны только для конкретной компоненты, в имени файла нужно указывать module, например:

**Navbar.module.css**

При этом все уникальные className, которые генерирует React, содержатся в объекте **classes** в виде:

(чтобы увидеть этот объект, его можно вывести в console.log(classes) )

{

’именование классов, которое я придумал’ : ’именование классов, которые сгенерировал React’,

’именование классов, которое я придумал’ : ’именование классов, которые сгенерировал React’,

…

}

Поэтому, в файле компонента пишем следующее:

- Для импорта объекта стилей:

import classes from './Profile.module.css'**;**

- Для присваивания классов в className вносим следующие значения (classes.posts, например):

export function Profile() {  
 return (  
 <div className="content">  
 <div>  
 <img  
 src="https://i.siteapi.org/no2IMQtVA-Jko-HOxCRaQpEC3FE=/fit-in/900x1000/center/top/filters:format(png)/5045b586a05f134.s.siteapi.org/img/d35c0d1ac7c8dd12e1e4146f14df612b9c31a4a3.jpg"  
 alt="nature"/>  
 </div>  
 <div>Ava + description</div>  
 <div>My posts  
 <div>New post</div>  
 <div className={classes.posts}>  
 <div className={classes.item}>Post1</div>  
 <div className={classes.item}>Post2</div>  
 </div>  
 </div>  
 </div>  
 )  
}

Чтобы дать тегу несколько классов, используем ШАБЛОННЫЕ СТРОКИ:

**className = {`${classes.nav} ${classes.active}`}**

Для аналога стилей CSS:

**.nav . active {**

**…**

**}**

Т.к. при определение классов мы обращаемся к свойствам объекта classes (можно при импорте назвать его по-другому), то мы не можем использовать в именах классов дефис. Можем, если только будем использовать другой синтаксис типа classNmae={classes.[“header-rm”]} вместо {classes.header}. **Поэтому, в именах классов ДЕФИС НЕ ИСПОЛЬЗУЕМ, можно camelCase, можно нижнее подчеркивание\_\_**

15. Структура папок, новые компоненты, улучшаем CSS

Сгруппируем в отдельные папки файл компонента и файл стилей.

Дробим наши полученные компоненты.

В папке исходного компонента создаем папку с именем компонента следующе уровня, вместе с ним и стили css.

Так мы можем использовать одно из главных преимуществ компонент, мы можем их потом дублировать тегами!

16. PROPS

Компонент - это функция, возвращающая разметку JSX

она же

Компонент - это тег

в React функция вызывается с параметром props

export function Header(props) {  
 return (  
 <header className={classes.header}>  
 <img  
 src="https://www.phpro.be/uploads/media/sulu-100x100/00/440-react%404x.png?v=2-0"  
 alt="logo"/>  
 </header>  
 )  
}

React всегда вкидывает объект в функцию (компоненту)

Когда мы указываем компоненту без аттрибутов props, в компоненту приходит пустой оъект.

Компоненту ИЗВНЕ мы настраиваем с помощью атрибутов.

Создадим атрибут компоненту <Header/>

<Header name="Dima K"/>

Тогда React автоматически создаст props объект и отрисовывается будет этот компонент уже с объектом. Т.е. компонент как функция будет иметь доступ к объекту как будто аргумент у функции.

{  
 name: "Dima K"  
}

Как использовать объект props в компоненте (функции) мы прописываем в фигурных скобках {}

export function Header(props) {  
 return (  
 <header className={classes.header}>  
 <img  
 src="https://www.phpro.be/uploads/media/sulu-100x100/00/440-react%404x.png?v=2-0"  
 alt="logo"/>  
 </header>  
 <span>{props.name}</span>  
 )  
}

18. Страница Dialogs

<App/> -это функция, которая возвращает JSX и принимает параметры props.

Один из принципов функционального программирования - функция должна возвращать какое-то значение, основываясь только на входящих данных (предсказуемое поведение)