React: TypeScript, Styles

AIT
Hi-tech team

НАШИ ПРАВИЛА

Включенная камера

Вопросы по поднятой руке

Не перебиваем друг друга

Все вопросы, не связанные с тематикой курса (орг-вопросы и т. д.), должны быть направлены куратору

Подготовьте свое рабочее окружение для возможной демонстрации экрана (закройте лишние соцсети и прочие приложения)

Повторим;)

- Что такое TypeScript?
- Где нужно указать тип переменной при её объявлении?
- Какие есть возможности типизации массива?
- Какие есть возможности типизации объекта?
- Где можно типизировать значение, которое возвращает функция
- Что такое hooks?

ЦЕЛЬ

Изучить тип enum и generic

ПЛАН ЗАНЯТИЯ

1. Дженерики в TypeScript
2. Тип enum

Generic Types

Дженерики (Generic Types):

Дженерики (generics) предоставляют средства для создания обобщенных типов и функций. Это позволяет писать более универсальный код, который может работать с различными типами данных, не теряя статической типизации

Пример

```
// Обобщенный тип для массива любого типа
type MyArray<T> = T[];

// Пример использования
const stringArray: MyArray<string> = ['a', 'b', 'c'];
const numberArray: MyArray<number> = [1, 2, 3];
```

Дженерики (Generic Types):

В дженериках можно задавать Default значения

Пример

```
// Обобщенный тип с default значением
type MyDefaultArray<T = number> = T[];

// Пример использования
const numberArray: MyDefaultArray = [1, 2, 3];
const stringArray: MyDefaultArray<string> = ['a', 'b', 'c'];
```

Дженерики (Generic Types):

Дженерики можно использовать в интерфейсах

Пример

```
// Обобщенный интерфейс для объекта с одним свойством
interface KeyValue<T> {
  key: string;
  value: T;
}

// Пример использования
const stringKeyValue: KeyValue<string> = { key: 'name', value: 'John' };
const numberKeyValue: KeyValue<number> = { key: 'age', value: 30 };
```

Enum type

enum

Тип enum или перечисление позволяет определить набор именнованных констант, которые описывают определенные состояния.

Для определения перечисления применяется ключевое слово enum. Например, объявим следующее перечисление:

```
enum Season { Winter, Spring, Summer, Autumn };
```

Перечисление называется Season и имеет четыре элемента. Теперь используем перечисление:

```
enum Season { Winter, Spring, Summer, Autumn };

let current: Season = Season.Summer;

console.log(current);       // 2
```

enum

Числовые перечисления

По умолчанию константы перечисления, как в примере выше, представляют числовые значения. То есть это так называемое числовое перечисление, в котором каждой константе сопоставляется числовое значение.

```
1  enum Season { Winter, Spring, Summer, Autumn };
```

Эквивалентно

```
1  enum Season { Winter=0, Spring=1, Summer=2, Autumn=3 };
```

enum

Числовые перечисления

Хотя мы можем явным образом переопределить эти значения. Так, мы можем задать значение одной константы, тогда значения следующих констант будет увеличиваться на единицу:

```
1  enum Season { Winter=5, Spring, Summer, Autumn };          // 5, 6, 7, 8
```

Либо можно каждой константе задать свое значение:

```
1  enum Season { Winter=4, Spring=8, Summer=16, Autumn=32 };   // 4, 8, 16, 32
```

enum

Строковые перечисления

Кроме числовых перечислений в TypeScript есть строковые перечисления, константы которых принимают строковые значения:

```
enum Season {
    Winter = "Зима",
    Spring = "Весна",
    Summer = "Лето",
    Autumn = "Осень"
};
var current: Season = Season.Summer;
console.log(current);   // Лето
```

Также можно определять смешанные перечисления, константы которых могут числа и строки.

Styling Components

1 подход

Обычный CSS

Написание стилей происходит в отдельных CSS файлах и затем они импортируются в ваши компоненты

```css
/* styles.css */
.myComponent {
  color: blue;
  font-size: 16px;
}
```

css файл

```jsx
// MyComponent.js
import React from 'react';
import './styles.css'; // Импорт стилей

const MyComponent = () => {
  return <div className="myComponent">Пример компонента</div>;
};

export default MyComponent;
```

react компонент

2 подход

Inline Styles:

Стили указываются напрямую внутри JSX элемента.

```jsx
const MyComponent = () => {
  return <div style={{ color: 'blue', fontSize: '16px' }}>Пример компонента</div>;
};
```

С помощью создания отдельной переменной

```jsx
const MyComponent = () => {
  const style = {
    color: 'blue',
    fontSize: '16px',
  };

  return <div style={style}>Пример компонента</div>;
};
```

3 подход

CSS-in-JS библиотеки, такой как emotion

Она предоставляет множество возможностей для создания стилей, включая локальную область видимости и использование динамических стилей.

Установка

Для начала, установите emotion в вашем проекте с помощью npm

```
npm install --save @emotion/react
```

```
npm install --save @emotion/styled
```

Ваша новая IT-профессия – Ваш новый уровень жизни

Программирование с нуля в
немецкой школе AIT TR GmbH

AIT
Hi-tech team