

## РАЗЛИЧИЯ МЕЖДУ TYPE И INTERFACE

Две этих сущности очень похожи, но есть и отличия, которые определяют что именно использовать в разных ситуациях

👉 Разница в синтаксисе создания новых type и interface:

```
1 interface IConfigWithRole extends IConfig, IRole {}
```

Оператор extends,  
внутри скобок можно добавить новые свойства

```
1 type ConfigWithRole = Config & Role;
```

Оператор intersection

👉 Разница в синтаксисе расширения существующих type и interface:

```
1 interface IConfig {  
2     protocol: "http" | "https";  
3     port: 3000 | 3001;  
4 }  
5  
6 interface IConfig {  
7     date: Date  
8 }
```

Допустимо добавлять новые поля  
в существующий интерфейс

```
1 type Config = {  
2     protocol: "http" | "https";  
3     port: 3000 | 3001;  
4 };  
5  
6 type Config = {  
7     date: Date  
8 }
```

**Ошибка!**  
Дублирование имени дает ошибку

Учтите, что расширение интерфейсов таким образом - **плохая практика**. Создавайте их сразу полноценными, а если необходимо внести дополнения - делайте их сразу в строках, где он объявлен.

**Исключение** - это когда приходится расширять сторонний интерфейс, который вы импортировали себе в код и не имеете доступа к оригинальной сущности.

👉 В интерфейсы можно помещать **только объекты**, в type допустимы **литералы**:

```
1 interface IProtocol "http" | "https";
2
3 interface IConfig {
4     protocol: "http" | "https";
5     port: 3000 | 3001;
6 }
```

В интерфейс поместить литерал не выйдет

```
1 type Protocol = "http" | "https";
2
3 type Config = {
4     protocol: Protocol;
5     port: 3000 | 3001;
6 };
```

В type помещен строковый литерал

На основании этих отличий мы и выбираем сущность для работы:

- ! Если вам нужен примитивный тип в качестве псевдонима – то тут **только типы**
- ! Если вы откуда-то берете готовый интерфейс и его нужно расширить – **это интерфейсы**
- ! Если же вы просто работаете с объектами, то конкретной разницы не будет

**Но чаще всего для работы с объектами используются интерфейсы!**  
Это связано и с дополнением сторонних интерфейсов, и с работой с классами, и с тем, что так исторически сложилось в программировании

Советую использовать именно их. Документация так же вам об этом скажет:

*If you would like a heuristic, use interface until you need to use features from type.*