

# TypeScript на максималках

Константин Логиновских  
Cloud.ru

# О чем поговорим?

# О чем поговорим?

## Как прокачать работу с урлами

# О чем поговорим?

Как прокачать работу с урлами

Как прокачать работу с конфигами

# О чем поговорим?

Как прокачать работу с урлами

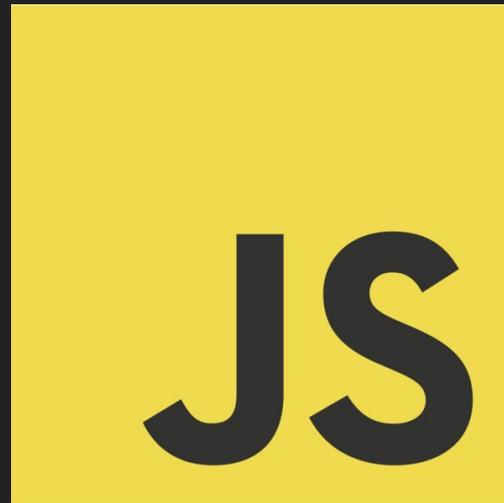
Как прокачать работу с конфигами

Какие есть еще типы

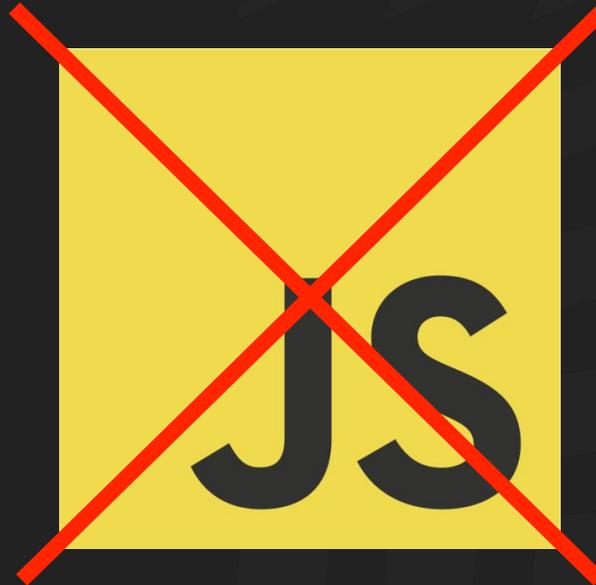
# Вдохновить на исследование TS

# О чем говорить не будем

# О чем говорить не будем



# О чем говорить не будем



## Ведущий разработчик Cloud.ru

Техлид внутреннего проекта

Вдохновлен TypeScript'ом



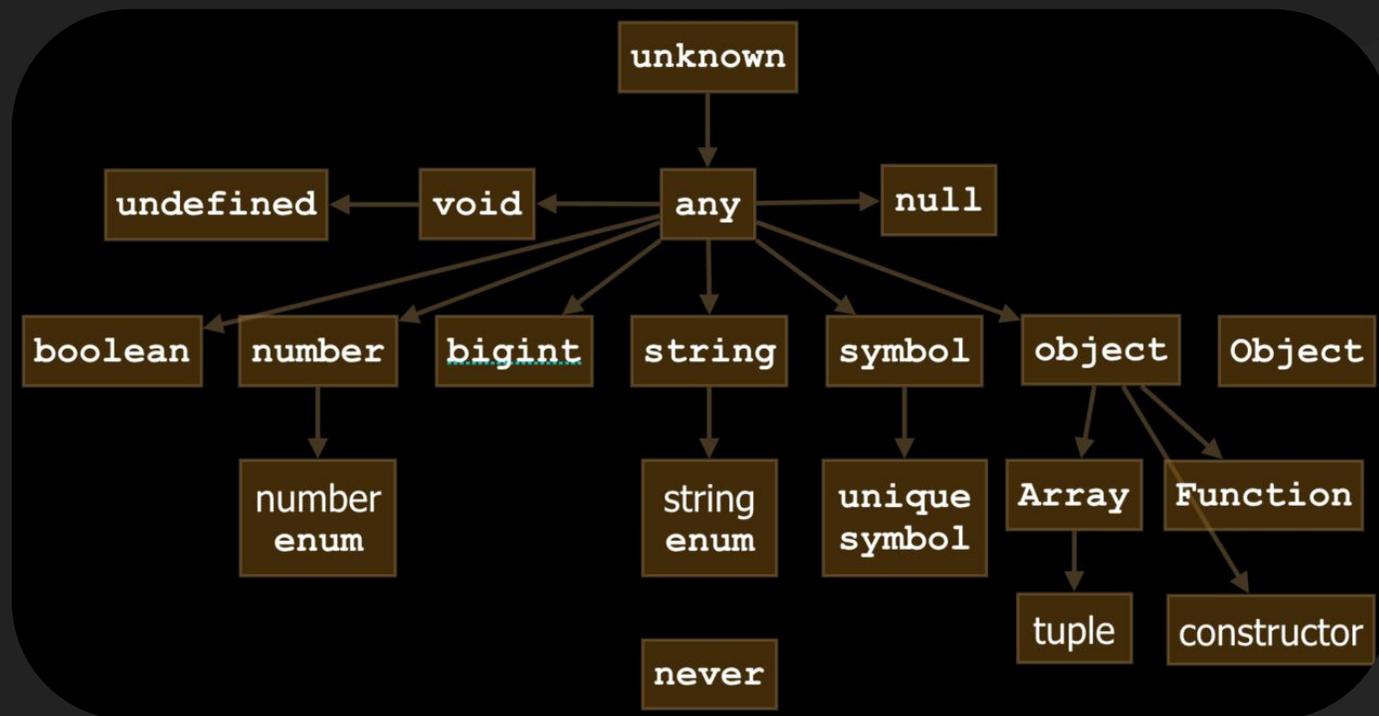
# План изучения TS

# План изучения TS

Выучить все примитивы

# План изучения TS

Выучить все примитивы



# План изучения TS

Выучить все примитивы

Разобраться с операторами

# План изучения TS

Выучить все примитивы

Разобраться с операторами

Познать базовые утилитарные типы

# План изучения TS

Выучить все примитивы

Разобраться с операторами

Познать базовые утилитарные типы

Научиться все это собирать

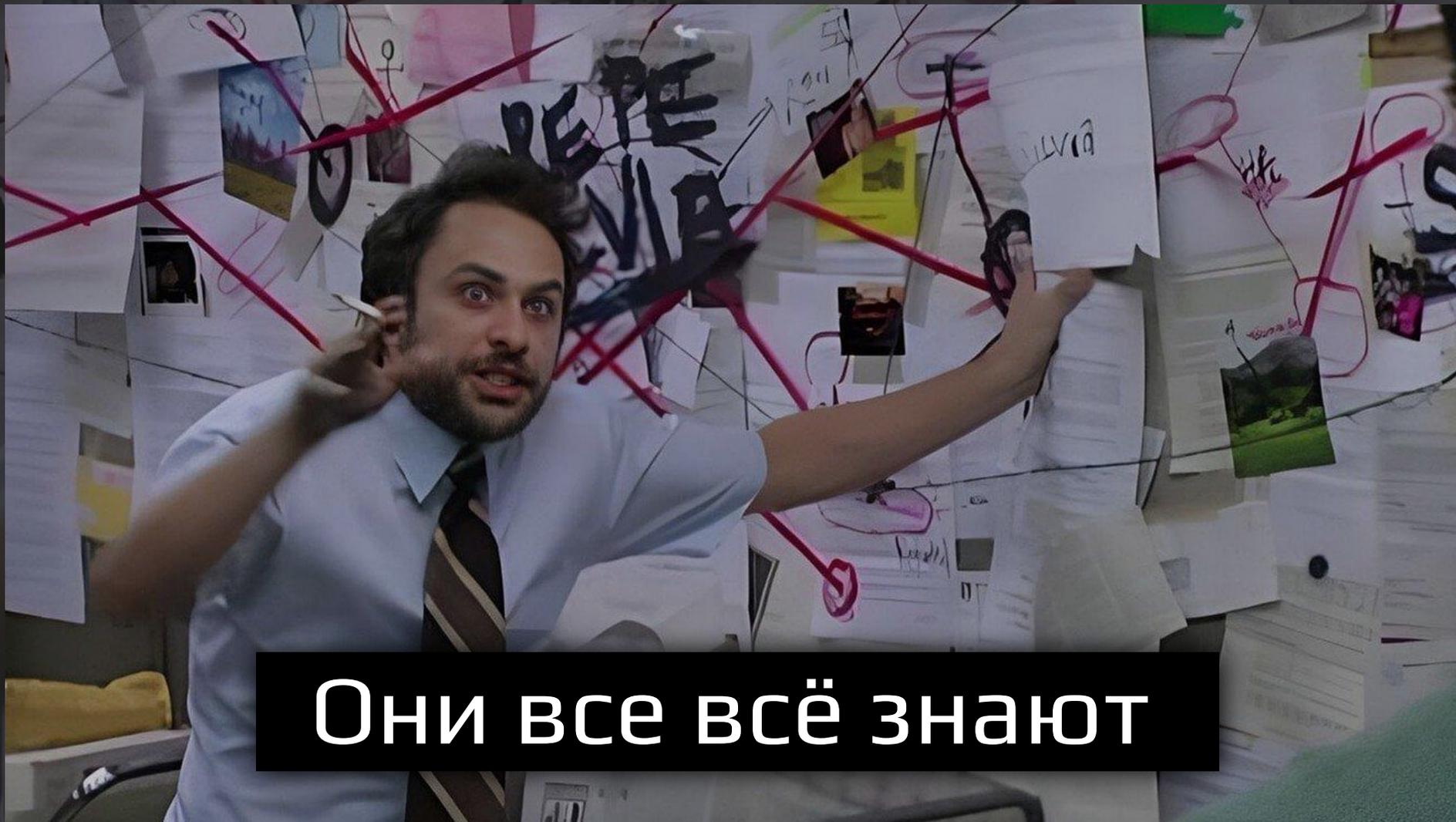
TypeScript 

# TypeChallenges



# TypeChallenges

# Почему про это молчат



Они все всё знают

# Прошу всех встать

# Кто НЕ писал на Typescript

# Awaited<T>

type MyAwaited<T> = ?

MyAwaited<Promise<Smth>> = Smth

type AwaitedAll<T> = ?

AwaitedAll<[Promise<Smth>]> = [Smth]

# infer



infer

# infer

# Разберем на пальцах

ЗАКАЗЧИК: 

ЭСКИЗ: 

ЛАК: 

ЗАКАЗЧИК: 

ЭСКИЗ: 

ЛАК: 

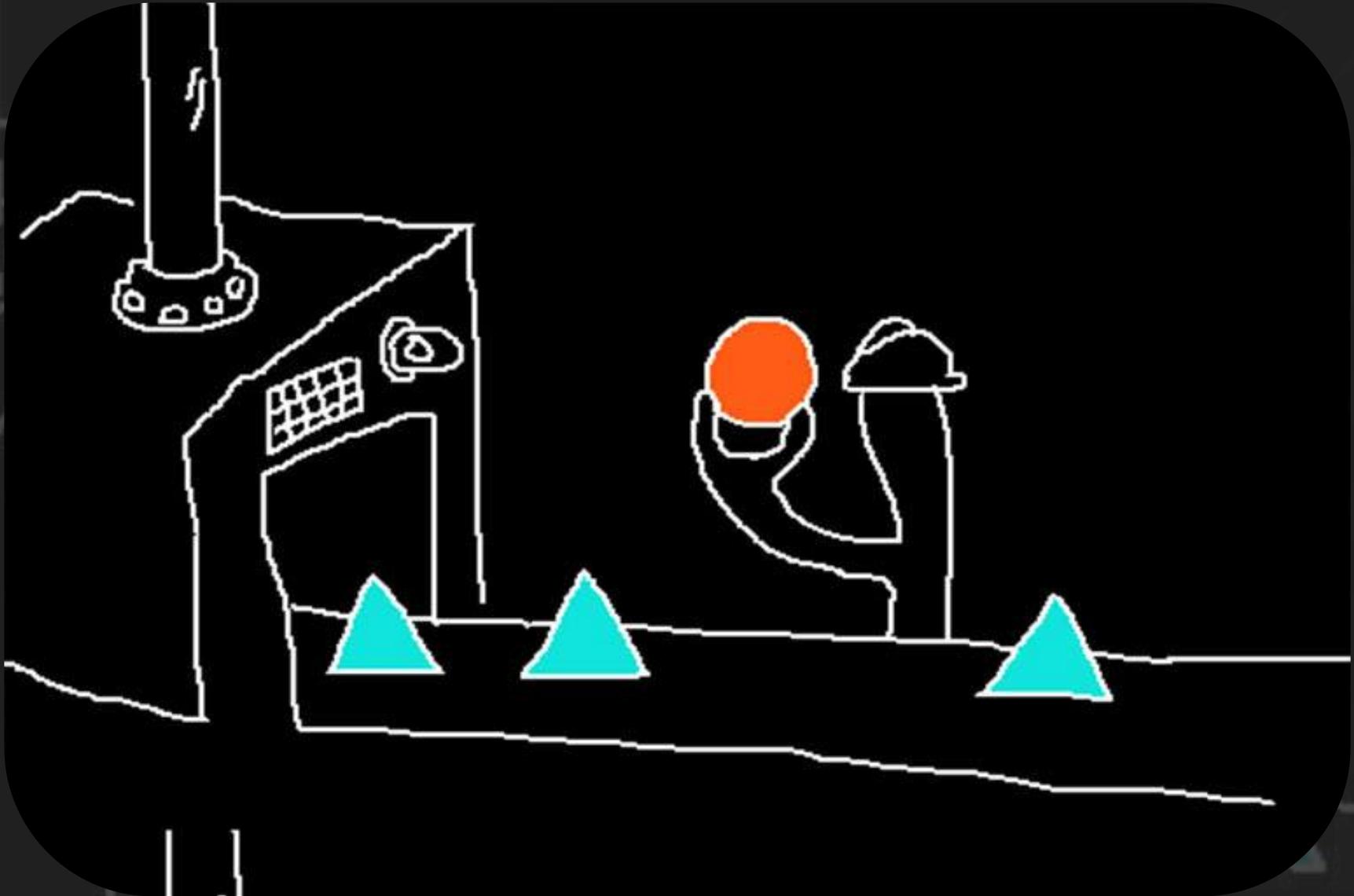


ЗАКАЗЧИК: 

ЭСКИЗ: 

ЛАК: 





ЗАКАЗЧИК: 

ЭСКИЗ: 

ЛАК: 

ЗАКАЗЧИК: 

ЭСКИЗ: 

ЛАК: 



...что-то ненужное

Эскиз: форма/цвет

...что-то ненужное

форма/цвет

ЗАКАЗЧИК: 

ЭСКИЗ: 

ЛАК: 



ЗАКАЗЧИК: любой  
ЭСКИЗ: **infer Shape**  
ЛАК: любой

**Shape**

ЗАКАЗЧИК: любой

ЭСКИЗ: **infer Shape**

ЛАК: любой

**Shape**

# Хватит на пальцах, дайте код!



```
1 type GetVariable<T> = T extends infer Var ? Var : never;
```



```
1 type GetVariable<T> = T extends infer Var ? Var : never;
```



```
1 type GetVariable<T> = T extends infer Var ? Var : never;  
2  
3 type CustomString = GetVariable<string> // string  
4 type CustomNumber = GetVariable<number> // number
```



```
1 type GetVariable<T> = T extends infer Var ? Var : never;  
2 type GetVariable<T> = T;
```



```
1 type FromArray<T> = ?
```

```
2
```

```
3 // FromArray<Array<SomeType>> => SomeType
```



```
1 type FromArray<T> = ?
```

```
2
```

```
3 // FromArray<Array<SomeType>> => SomeType
```



```
1 type FromArray<T> = T extends Array<infer U> ? U : never;
```

```
2
```

```
3
```

```
4
```



```
1 type FromArray<T> = T extends Array<infer U> ? U : never;  
2  
3 type CustomString = FromArray<Array<string>> // string  
4 type CustomNumber = FromArray<Array<number>> // number
```



```
1 type FromArray<T> = T extends Array<infer U> ? U : never;  
2  
3 type CustomString = FromArray<Array<string>> // string  
4 type CustomNumber = FromArray<Array<number>> // number
```



```
1 type FromArray<T> = T extends Array<infer U> ? U : never;  
2  
3 type CustomString = FromArray<Array<string>> // string  
4 type CustomNumber = FromArray<Array<number>> // number
```

# О чем поговорим?

Как прокачать работу с урлами

Как прокачать работу с конфигами

Какие еще есть кейсы

# О чем поговорим?

Как прокачать работу с урлами

Как прокачать работу с конфигами

Какие еще есть кейсы

# Подстановка параметров в путь



```
1 <Link to="/app/user/1234-4321" />
2
3 <Button onClick={() => navigateToUser(userId)} />
4
5 <a href={`/${BASE}/user/${userId}`} />
```



```
1 <Link to="/app/user/1234-4321" />
2
3 <Button onClick={() => navigateToUser(userId)} />
4
5 <a href={`/${BASE}/user/${userId}`} />
```



```
1 <Link to="/app/user/1234-4321" />
2
3 <Button onClick={() => navigateToUser(userId)} />
4
5 <a href={`/${BASE}/user/${userId}`} />
```



```
1 <Link to="/app/user/1234-4321" />
2
3 <Button onClick={() => navigateToUser(userId)} />
4
5 <a href={`/${BASE}/user/${userId}`} />
```

# Постановка задачи

Запросные урлы хранятся в переменных

# Постановка задачи

Запросные урлы хранятся в переменных



```
1  const resourceUrl = 'user/resource/id'
```

# Постановка задачи

Запросные урлы хранятся в переменных

Урл может содержать слоты для динамических полей

# Постановка задачи

Запросные урлы хранятся в переменных

Урл может содержать слоты для динамических полей



```
1 ' :user/resource/:id'
```

# Постановка задачи

Запросные урлы хранятся в переменных

Урл может содержать слоты для динамических полей



1

`:user/resource/:id'`

# Постановка задачи

Запросные урлы хранятся в переменных

Урл может содержать слоты для динамических полей

Нужна функция, которая заставит пользователя подставить все параметры

# Целевая картина



```
1  const resourceUrl = ':user/resource/:id'
```

# Целевая картина



```
1 const resourceUrl = ':user/resource/:id'
```



```
1 generatePath(resourceUrl, {});
```

# Целевая картина



```
1 const resourceUrl = ':user/resource/:id'
```



```
generatePath(resourceUrl, {});
```

 id

 user

 expand template

# Целевая картина



```
1 const resourceUrl = ':user/resource/:id'
```



```
generatePath(resourceUrl
```

```
{}) ;
```

```
id
```

```
user
```



```
1 function generatePath<T extends string>(path: T, params: Record<string, unknown>) {}
```



**Функцию писать, конечно  
же, никто не собирался**



```
1  function generatePath<T extends string>(
2      route: T,
3      params: ExtractPaths<T>
4  );
```



```
1  function generatePath<T extends string>(
2      route: T,
3      params: ExtractPaths<T>
4  );
```



```
1  function generatePath<T extends string>(
2      route: T,
3      params: ExtractPaths<T>
4  );
```



```
1 type ExtractPath<T extends string> = ?
```





```
1  type A = string;  
2  
3  type B = 'string';
```



```
1  type A = string;  
2  
3  type B = 'string';
```



```
1  type B = 'string';  
2  
3  type C = `Literal: ${B}`  
4  // 'Literal: string'
```



```
1 type ExtractPath<T extends string> = ?
```



```
1 type ExtractPath<T extends string> =  
2   T extends `:${infer Param}` ? Param : string;
```



```
1 type ExtractPath<T extends string> =  
2 T extends `:${infer Param}` ? Param : string;
```



```
1 type ExtractPath<T extends string> =  
2   T extends `${infer Param}` ? Param : string;
```



```
1 type ExtractPath<T extends string> =  
2   T extends `:${infer Param}` ? Param : string;
```



```
1 ExtractPath<'user'> => 'user'  
2 ExtractPath<'user'> => string
```



```
1 type ExtractPath<T extends string> =  
2   T extends `:${infer Param}` ? Param : string;  
3  
4 type ExtractPaths<T extends string> = ?
```



```
1 type ExtractPaths<T extends string> =  
2   T extends `${infer Left}/${infer Right}`
```



```
1 type ExtractPaths<T extends string> =  
2   T extends `${infer Left}/${infer Right}`
```



```
1 type ExtractPaths<T extends string> =  
2   T extends `${infer Left}/${infer Right}`  
3   ? ExtractPaths<Left> | ExtractPaths<Right>  
4   : ExtractPath<T>;
```



```
1 type ExtractPaths<T extends string> =  
2   T extends `${infer Left}/${infer Right}`  
3     ? ExtractPaths<Left> | ExtractPaths<Right>  
4     : ExtractPath<T>;
```



```
1 type ExtractPaths<T extends string> =  
2   T extends `${infer Left}/${infer Right}`  
3     ? ExtractPaths<Left> | ExtractPaths<Right>  
4     : ExtractPath<T>;
```



```
1 type ExtractPaths<T extends string> =  
2   T extends `${infer Left}/${infer Right}`  
3   ? ExtractPaths<Left> | ExtractPaths<Right>  
4   : ExtractPath<T>;
```



```
1 type ExtractPaths<T extends string> =  
2   T extends `${infer Left}/${infer Right}`  
3   ? ExtractPaths<Left> | ExtractPaths<Right>  
4   : ExtractPath<T>;
```



```
1 type ExtractPaths<T extends string> =  
2   T extends `${infer Left}/${infer Right}`  
3     ? ExtractPaths<Left> | ExtractPaths<Right>  
4     : ExtractPath<T>;
```



```
1 'user/:id/resources'  
2  
3  
4
```



```
1 type ExtractPaths<T extends string> =  
2   T extends `${infer Left}/${infer Right}`  
3   ? ExtractPaths<Left> | ExtractPaths<Right>  
4   : ExtractPath<T>;
```



```
1 'user/:id/resources'  
2 => ExtractPaths<'user'> | ExtractPaths<':id/resources'>  
3  
4
```



```
1 type ExtractPaths<T extends string> =  
2   T extends `${infer Left}/${infer Right}`  
3   ? ExtractPaths<Left> | ExtractPaths<Right>  
4   : ExtractPath<T>;
```



```
1 'user/:id/resources'  
2 => ExtractPaths<'user'> | ExtractPaths<' :id/resources '>  
3 => ExtractPath<'user'>  
4
```



```
1 type ExtractPaths<T extends string> =  
2   T extends `${infer Left}/${infer Right}`  
3   ? ExtractPaths<Left> | ExtractPaths<Right>  
4   : ExtractPath<T>;
```



```
1 'user/:id/resources'  
2 => ExtractPaths<'user'> | ExtractPaths<':id/resources'>  
3 => ExtractPath<'user'> | ExtractPaths<':id'> | ExtractPaths<'resources'>  
4
```



```
1 type ExtractPaths<T extends string> =  
2   T extends `${infer Left}/${infer Right}`  
3   ? ExtractPaths<Left> | ExtractPaths<Right>  
4   : ExtractPath<T>;
```



```
1 'user/:id/resources'  
2 => ExtractPaths<'user'> | ExtractPaths<' :id/resources '>  
3 => ExtractPath<'user'> | ExtractPaths<' :id '> | ExtractPaths<'resources '>  
4 => ExtractPath<'user'> | ExtractPath<' :id '> | ExtractPath<'resources '>
```



```
1 type PathParams<T extends string> = Record<ExtractPaths<T>, string>
```



```
1 type ExtractPath<T extends string> = T extends `:${infer Param}` ? Param : string;
2
3 type ExtractPaths<T extends string> = T extends `${infer Left}/${infer Right}`
4   ? ExtractPaths<Left> | ExtractPaths<Right>
5   : ExtractPath<T>;
6
7 function generatePath<T extends string>(_path: T, _params: ExtractPaths<T>) {...}
```



```
1  ':user/resource/:id'
```

```
2
```

```
3
```



```
1  ':user/resource/:id'  
2  => ExtractPath<':user'> | ExtractPath<'resource'> | ExtractPath<':id'>  
3
```



```
1  ':user/resource/:id'  
2  => ExtractPath<':user'> | ExtractPath<'resource'> | ExtractPath<':id'>  
3  => 'user' | string | 'id'
```



```
1  ':user/resource/:id'  
2  => ExtractPath<':user'> | ExtractPath<'resource'> | ExtractPath<':id'>  
3  => 'user' | string | 'id'  
4  => string
```



```
1 type ExtractPath<T extends string> = T extends `:${infer Param}` ? Param : string;
2
3 type ExtractPaths<T extends string> = T extends `${infer Left}/${infer Right}`
4   ? ExtractPaths<Left> | ExtractPaths<Right>
5   : ExtractPath<T>;
6
7 function generatePath<T extends string>(_path: T, _params: ExtractPaths<T>) {...}
```



```
1 type ExtractPath<T extends string> = T extends `:${infer Param}` ? Param : never;
2
3 type ExtractPaths<T extends string> =
4   T extends `${infer Left}/${infer Right}`
5     ? ExtractPaths<Left> | ExtractPaths<Right>
6     : ExtractPath<T>;
7
8 type PathParams<T extends string> = Record<ExtractPaths<T>, string>
9
10 function generatePath<T extends string>(_path: T, _params: PathParams<T>) {...}
```



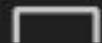
```
1 type ExtractPath<T extends string> = T extends `:${infer Param}` ? Param : never;
2
3 type ExtractPaths<T extends string> =
4   T extends `${infer Left}/${infer Right}`
5     ? ExtractPaths<Left> | ExtractPaths<Right>
6     : ExtractPath<T>;
7
8 type PathParams<T extends string> = Record<ExtractPaths<T>, string>
9
10 function generatePath<T extends string>(_path: T, _params: PathParams<T>) {...}
```



```
generatePath(resourceUrl, {});
```

 id

 user

 expand-template

# Полезные доработки: читаемость типа

Аргумент типа "{}" нельзя назначить параметру типа "PathParams<":user/resource/:id">".  
В типе "{}" отсутствуют следующие свойства из типа "PathParams<":user/resource/:id">": user,  
id ts(2345)

[Просмотреть проблему \(\F8\)](#) [Быстрое исправление... \(\.\)](#)

```
generatePath(resourceUrl, {});
```

Аргумент типа "{}" нельзя назначить параметру типа "PathParams<":user/resource/:id">".

~~В типе "[]" отсутствуют следующие свойства из типа "PathParams<":user/resource/:id">": user,  
id ts(2345)~~

[Просмотреть проблему \(\F8\)](#) [Быстрое исправление... \(\.\)](#)

```
generatePath(resourceUrl, {});
```



```
1 type Prettyfy<T> = T extends Record<string, unknown> ? {  
2   [K in keyof T]: Prettyfy<T[K]>;  
3 } : T;
```



```
1 type Prettyfy<T> = T extends Record<string, unknown> ? {  
2   [K in keyof T]: Prettyfy<T[K]>;  
3 } : T;
```

Аргумент типа "{}" нельзя назначить параметру типа "PathParams<":user/resource/:id">".  
В типе "{}" отсутствуют следующие свойства из типа "PathParams<":user/resource/:id">": user,  
id ts(2345)

[Просмотреть проблему \(\F8\)](#) [Быстрое исправление... \(\.\)](#)

```
generatePath(resourceUrl, {});
```

Аргумент типа "{}" нельзя назначить параметру типа "{ user: string; id: string; }".

В типе "{}" отсутствуют следующие свойства из типа "{ user: string; id: string; }": user, id ts(2345)

[Просмотреть проблему \(↗\)](#) [Быстрое исправление... \(↗\)](#)

```
generatePath(resourceUrl, {});
```

# Полезные доработки: опциональный аргумент



```
1  const userUrl = 'user/me';  
2  
3  generatePath(userUrl);
```



```
1  const userUrl = 'user/me';  
2  
3  generatePath(userUrl);
```

Ожидалось аргументов: 2, получено: 1. ts(2554)

protected.tsx(24, 50): Не указан аргумент для "params".

```
function generatePath<"user/me">(path: "user/me", params: {}):  
string
```



```
1  type OptionalType = unknown;  
2  
3  function withoutArgument(arg: OptionalType) {}  
4  
5  withoutArgument();  
6  // Ожидалось аргументов: 1, получено: 0.
```



```
1 type OptionalType = unknown;  
2  
3 function withoutArgument(arg: OptionalType) {}  
4  
5 withoutArgument();  
6 // Ожидалось аргументов: 1, получено: 0.
```



```
1 type OptionalType = unknown;
2
3 function withoutArgument(arg: OptionalType) {}
4
5 withoutArgument();
6 // Ожидалось аргументов: 1, получено: 0.
```



```
1  type OptionalType = unknown;
2
3  function withoutArgument(arg: OptionalType) {}
4
5  withoutArgument();
6  // Ожидалось аргументов: 1, получено: 0.
```



```
1 type OptionalType = unknown;  
2  
3 function withoutArgument(arg: OptionalType) {}  
4  
5 withoutArgument();  
6 // Ожидалось аргументов: 1, получено: 0.
```



```
1  type OptionalType = undefined;  
2  
3  function withoutArgument(arg: OptionalType) {}  
4  
5  withoutArgument();  
6
```



```
1  type OptionalType = undefined;
2
3  function withoutArgument(arg: OptionalType) {}
4
5  withoutArgument();
6  // Ожидалось аргументов: 1, получено: 0.
```



```
1  type OptionalType = never;  
2  
3  function withoutArgument(arg: OptionalType) {}  
4  
5  withoutArgument();  
6  // Ожидалось аргументов: 1, получено: 0.
```



```
1  type OptionalType = void;
2
3  function withoutArgument(arg: OptionalType) {}
4
5  withoutArgument();
6  // ok?
```



```
1  type OptionalType = void;
2
3  function withoutArgument(arg: OptionalType) {}
4
5  withoutArgument(returnsVoid());
6  // (J ° □ °) J (LL)
```



```
1 declare function wrapper(fn: () => void): typeof fn
2
3 const returnsString = (): string => '42';
4
5 const returnsVoid = wrapper(returnsString)
```



```
1  function withoutArguments(): void;
2  function withoutArguments(arg: string): void;
3  function withoutArguments(arg?: string): void {};
4
5  withoutArguments('1:2'); // ok
6  withoutArguments(); // ok
```



```
1  type OptionalType = [];  
2  
3  function withoutArgument(...arg: OptionalType) {}  
4  
5  withoutArgument();
```



```
1  type OptionalType = [];  
2  
3  function withoutArgument(...arg: OptionalType) {}  
4  
5  withoutArgument();
```



```
1 function generatePath<T extends string>(...params: Args<T>): string {...}
```



```
1 type Args<T> = Record<string, never> extends PathParams<T>
2   ? [path: T]
3   : [path: T, params: PathParams<T> ];
```



```
1 type Args<T> = Record<string, never> extends PathParams<T>
2   ? [path: T]
3   : [path: T, params: PathParams<T> ];
```



```
1 type Args<T> = Record<string, never> extends {}  
2   ? [path: T]  
3   : [path: T, params: PathParams<T> ];
```



```
1 type Args<T> = Record<string, never> extends {}  
2   ? [path: T]  
3   : [path: T, params: PathParams<T> ];
```



```
1 type Args<T> = Record<string, never> extends { id: string }  
2   ? [path: T]  
3   : [path: T, params: PathParams<T> ];
```



```
1 type Args<T> = Record<string, never> extends { id: string }  
2   ? [path: T]  
3   : [path: T, params: PathParams<T> ];
```

```
const resourceUrl = ':user/resource/:id';  
generatePath(resourceUrl);
```

```
const userUrl = 'user/me';  
generatePath(userUrl);
```

```
const resourceUrl = ':user/resource/:id';  
generatePath(resourceUrl, { id: '2' });
```

```
const userUrl = 'user/me';  
generatePath(userUrl);
```



```
1  export function Link({ url, params, ...props }: LinkProps) {  
2      const href = params ? generatePath(url, params) : url;  
3  
4      return <RouterLink href={href} {...props} />  
5  }
```

# О чем поговорим?

Как прокачать работу с урлами

Как прокачать работу с конфигами

Какие еще есть кейсы

# О чем поговорим?

Как прокачать работу с урлами

Как прокачать работу с конфигами

Какие еще есть кейсы

# Идем вглубь

```
1  const ROUTE_CONFIG = {
2    url: '',
3    id: 'root',
4    children: [
5      {
6        url: ':user',
7        id: 'user',
8        children: [
9          {
10             url: 'resource',
11             id: 'resource',
12             children: [
13               { url: ':id' }
14             ]
15           }
16         ]
17       },
18       {
19         url: 'settings',
20         id: 'settings'
21       }
22     ]
23   }
```

```
1  const ROUTE_CONFIG = {
2    url: '',
3    id: 'root',
4    children: [
5      {
6        url: ':user',
7        id: 'user',
8        children: [
9          {
10           url: 'resource',
11           id: 'resource',
12           children: [
13             { url: ':id' }
14           ]
15         }
16       ]
17     },
18     {
19       url: 'settings',
20       id: 'settings'
21     }
22   ]
23 }
```

```
1  const paths = {
2    root: '/',
3    user: '/:user',
4    resource: '/:user/resource',
5    resourceId: '/:user/resource/:id',
6    settings: '/settings'
7  }
```

```
1  const ROUTE_CONFIG = {
2    url: '',
3    id: 'root',
4    children: [
5      {
6        url: ':user',
7        id: 'user',
8        children: [
9          {
10           url: 'resource',
11           id: 'resource',
12           children: [
13             { url: ':id' }
14           ]
15         }
16       ]
17     },
18     {
19       url: 'settings',
20       id: 'settings'
21     }
22   ]
23 }
```



```
1  const paths = {
2    root: '/',
3    user: '/:user',
4    resource: '/:user/resource',
5    resourceId: '/:user/resource/:id',
6    settings: '/settings'
7  }
```

```
1  const ROUTE_CONFIG = {
2    url: '',
3    id: 'root',
4    children: [
5      {
6        url: ':user',
7        id: 'user',
8        children: [
9          {
10             url: 'resource',
11             id: 'resource',
12             children: [
13               { url: ':id' }
14             ]
15           }
16         ]
17       },
18       {
19         url: 'settings',
20         id: 'settings'
21       }
22     ]
23   }
```

```

1  const ROUTE_CONFIG = {
2      url: '',
3      id: 'root',
4      children: [
5          {
6              url: ':user',
7              id: 'user',
8              children: [
9                  {
10                     url: 'resource',
11                     id: 'resource',
12                     children: [
13                         { url: ':id' }
14                     ]
15                 }
16             ]
17         },
18         {
19             url: 'settings',
20             id: 'settings'
21         }
22     ]
23 }

```

```

const ROUTE_CONFIG: {
    url: string;
    id: string;
    children: ({
        url: string;
        id: string;
        children: {
            url: string;
            id: string;
            children: {
                url: string;
            }[];
        }[];
    }[]);
} | {

```



```
1  const ROUTE_CONFIG = {  
2    url: '',  
3    id: 'root',  
4    children: [  
5      ...  
6    ]  
7  } as const;
```

```
1  const ROUTE_CONFIG = {
2    url: '',
3    id: 'root',
4    children: [
5      {
6        url: ':user',
7        id: 'user',
8        children: [
9          {
10           url: 'resource',
11           id: 'resource',
12           children: [
13             { url: ':id' }
14           ]
15         }
16       ]
17     },
18     {
19       url: 'settings',
20       id: 'settings'
21     }
22   ]
23 } as const;
```

```
const ROUTE_CONFIG: {
  readonly url: "";
  readonly id: "root";
  readonly children: readonly [{
    readonly url: ":user";
    readonly id: "user";
    readonly children: readonly [{
      readonly url: "resource";
      readonly id: "resource";
      readonly children: readonly [{
        readonly url: ":id";
      }];
    }];
  }];
}
```

```
1  const ROUTE_CONFIG = {
2    url: '',
3    id: 'root',
4    children: [
5      {
6        url: ':user',
7        id: 'user',
8        children: [
9          {
10           url: 'resource',
11           id: 'resource',
12           children: [
13             { url: ':id' }
14           ]
15         }
16       ]
17     },
18     {
19       url: 'settings',
20       id: 'settings'
21     }
22   ]
23 } as const;
```

```
1  const ROUTE_CONFIG = {
2    url: '',
3    id: 'root',
4    children: [
5      {
6        url: ':user',
7        id: 'user',
8        children: [
9          {
10           url: 'resource',
11           id: 'resource',
12           children: [
13             { url: ':id' }
14           ]
15         }
16       ]
17     },
18     {
19       url: 'settings',
20       id: 'settings'
21     }
22   ]
23 } as const;
```

```
1  type RouteConfig = {
2    id: string;
3    url: string;
4    children?: RouteConfig[];
5  }
```



```
1  const ROUTE_CONFIG: RouteConfig = {  
2    url: '',  
3    id: 'root',
```

```

1  const ROUTE_CONFIG = {
2    url: '',
3    id: 'root',
4    children: [
5      {
6        url: ':user',
7        id: 'user',
8        children: [
9          {
10         url: 'resource',
11         id: 'resource',
12         children: [
13           { url: ':id' }
14         ]
15       }
16     ]
17   },
18   {
19     url: 'settings',
20     id: 'settings'
21   }
22 ]
23 } as const;

```

```

const ROUTE_CONFIG: {
  readonly url: "";
  readonly id: "root";
  readonly children: readonly [{
    readonly url: ":user";
    readonly id: "user";
    readonly children: readonly [{
      readonly url: "resource";
      readonly id: "resource";
      readonly children: readonly [{
        readonly url: ":id";
      }];
    }];
  }];
}

```

```

1  const ROUTE_CONFIG = {
2    url: '',
3    id: 'root',
4    children: [
5      {
6        url: ':user',
7        id: 'user',
8        children: [
9          {
10         url: 'resource',
11         id: 'resource',
12         children: [
13           { url: ':id' }
14         ]
15       }
16     ]
17   },
18   {
19     url: 'settings',
20     id: 'settings'
21   }
22 ]
23 } as const;

```

```

const ROUTE_CONFIG: {
  readonly url: "";
  readonly id: "root";
  readonly children: readonly [{
    readonly url: ":user";

```

**const ROUTE\_CONFIG: RouteConfig**

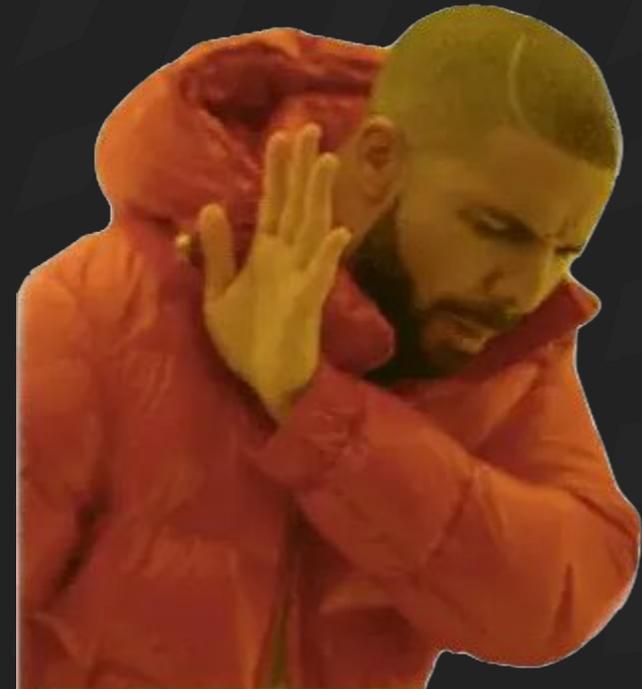
```

  readonly url: "resource";
  readonly id: "resource";
  readonly children: readonly [{
    readonly url: ":id";
  }];
}];
} {

```

```
1  type RouteConfig = {
2    id: string;
3    url: string;
4    children?: RouteConfig[];
5  }
6
7  interface InnerRouteConfig extends RouteConfig {
8    id: 'root'
9    url: '',
10   children: [
11     {
12       id: 'user',
13       url: 'user',
14     }
15   ]
16 }
17
18 const ROUTE_CONFIG: InnerRouteConfig = {...}
```

```
1 type RouteConfig = {
2   id: string;
3   url: string;
4   children?: RouteConfig[];
5 }
6
7 interface InnerRouteConfig extends RouteConfig {
8   id: 'root'
9   url: '',
10  children: [
11    {
12      id: 'user',
13      url: 'user',
14    }
15  ]
16 }
17
18 const ROUTE_CONFIG: InnerRouteConfig = {...}
```





```
1  const ROUTE_CONFIG: RouteConfig = {  
2    url: '',  
3    id: 'root',
```



```
1  const ROUTE_CONFIG = {  
2    url: '',  
3    id: 'root',  
4    children: [...]  
5  } as const satisfies RouteConfig;
```



```
1  const ROUTE_CONFIG = {  
2    url: '',  
3    id: 'root',  
4    children: [...]  
5  } as const satisfies RouteConfig;
```

```
1 type RouteConfig = {
2   id: string;
3   url: string;
4   children?: RouteConfig[];
5 }
6
7 interface InnerRouteConfig extends RouteConfig {
8   id: 'root'
9   url: '',
10  children: [
11    {
12      id: 'user',
13      url: 'user',
14    }
15  ]
16 }
17
18 const ROUTE_CONFIG: InnerRouteConfig = {...}
```

```
1 const ROUTE_CONFIG = {
2   url: '',
3   id: 'root',
4   children: [...]
5 } as const satisfies RouteConfig;
```

```
1 type RouteConfig = {
2   id: string;
3   url: string;
4   children?: RouteConfig[];
5 }
6
7 interface InnerRouteConfig extends RouteConfig {
8   id: 'root'
9   url: '',
10  children: [
11    {
12      id: 'user',
13      url: 'user',
14    }
15  ]
16 }
17
18 const ROUTE_CONFIG: InnerRouteConfig = {...}
```

```
1 const ROUTE_CONFIG = {
2   url: '',
3   id: 'root',
4   children: [...]} as const satisfies RouteConfig;
```

```
1 type RouteConfig = {
2   id: string;
3   url: string;
4   children?: RouteConfig[];
5 }
6
7 interface InnerRouteConfig extends RouteConfig {
8   id: 'root'
9   url: '',
10  children: [
11    {
12      id: 'user',
13      url: 'user',
14    }
15  ]
16 }
17
18 const ROUTE_CONFIG: InnerRouteConfig = {...}
```

```
1 const ROUTE_CONFIG = {
2   url: '',
3   id: 'root',
4   children: [...]} as const satisfies RouteConfig;
```

```
1  const ROUTE_CONFIG = {
2    url: '',
3    id: 'root',
4    children: [
5      {
6        url: ':user',
7        id: 'user',
8        children: [
9          {
10         url: 'resource',
11         id: 'resource',
12         children: [
13           {
14             url: ':id',
15             id: 'resourceId',
16           },
17         ],
18       },
19     ],
20   },
21   {
22     url: 'settings',
23     id: 'settings',
24   },
25 ],
26 } as const satisfies RouteConfig;
```

```
1  const ROUTE_CONFIG = {
2    url: '',
3    id: 'root',
4    children: [
5      {
6        url: ':user',
7        id: 'user',
8        children: [
9          {
10           url: 'resource',
11           id: 'resource',
12           children: [
13             {
14               url: ':id',
15               id: 'resourceId',
16             },
17           ],
18         },
19       ],
20     },
21     {
22       url: 'settings',
23       id: 'settings',
24     },
25   ],
26 } as const satisfies RouteConfig;
```

```
children: [
  {
    url: ':id',
  },
],
```

```

1  const ROUTE_CONFIG = {
2    url: '',
3    id: 'root',
4    children: [
5      {
6        url: ':user',
7        id: 'user',
8        children: [
9          {
10         url: 'resource',
11         id: 'resource',
12         children: [
13           {
14             url: ':id',
15             id: 'resourceId',
16           },
17         ],
18       },
19     ],
20   },
21   {
22     url: 'settings',
23     id: 'settings',
24   },
25 ],
26 } as const satisfies RouteConfig;

```

```

children: [
  {
    url: ':id',
    id: 'resourceId',
    children: [
    ],
  },
],

```

```

const ROUTE_CONFIG: {
  readonly url: "";
  readonly id: "root";
  readonly children: [{
    readonly url: ":user";
    readonly id: "user";
    readonly children: [{
      readonly url: "resource";
      readonly id: "resource";
      readonly children: [{
        readonly url: ":id";
        readonly id: "resourceId";
      }];
    }];
  }];
};

```

```

1  const ROUTE_CONFIG = {
2    url: '',
3    id: 'root',
4    children: [
5      {
6        url: ':user',
7        id: 'user',
8        children: [
9          {
10         url: 'resource',
11         id: 'resource',
12         children: [
13           {
14             url: ':id',
15             id: 'resourceId',
16           },
17         ],
18       },
19     ],
20   },
21   {
22     url: 'settings',
23     id: 'settings',
24   },
25 ],
26 } as const satisfies RouteConfig;

```

```

children: [
  {
    url: ':id',
  },
],

```

```

const ROUTE_CONFIG: {
  readonly url: "";
  readonly id: "root";
  readonly children: [{
    readonly url: ":user";
    readonly id: "user";
    readonly children: [{
      readonly url: "resource";
      readonly id: "resource";
      readonly children: [{
        readonly url: ":id";
        readonly id: "resourceId";
      }];
    }];
  }];
}

```

```

11.

```





```
1  type GeneratePaths<T extends RouteConfig> = unknown;  
2  
3  type Paths = GeneratePaths<typeof ROUTE_CONFIG>; // unknown
```

# config

# config

# extract(config)

# config

extract(config)

...extract(config.children)

config

extract(config)

...extract(config.children)



```
1 type ExtractConfig<T extends RouteConfig, Base extends string = ''> =  
2   {[K in T['id']]: `${Base}/${T['url']}`}`}
```



```
1 type ExtractConfig<T extends RouteConfig, Base extends string = ''> =  
2   {[K in T['id']]: `${Base}/${T['url']}`}`}
```



```
1 type ExtractConfig<T extends RouteConfig, Base extends string = ''> =  
2   {[K in T['id']]: `${Base}/${T['url']}`} }
```



```
1 type ExtractConfig<T extends RouteConfig, Base extends string = ''> =  
2   {[K in T['id']]: `${Base}/${T['url']}`}
```



```
1 ExtractConfig<{id: 'someId'; url: 'some'}>  
2   => {someId: 'some'}
```

config

extract(config)

...extract(config.children)

config

extract(config)

...extract(config.children)



```
1 type ExtractConfig<T extends RouteConfig, Base extends string = ''> =  
2   {[K in T['id']]: `${Base}/${T['url']}`}  
3   & ExtractChildren<T['children'], `${Base}/${T['url']}`>  
4  
5 type ExtractChildren<T extends RouteConfig[], Base extends string = ''>  
6   = ?
```



```
1 type ExtractConfig<T extends RouteConfig, Base extends string = ''> =  
2   {[K in T['id']]: `${Base}/${T['url']}`}  
3   & ExtractChildren<T['children'], `${Base}/${T['url']}`>  
4  
5 type ExtractChildren<T extends RouteConfig[], Base extends string = ''>  
6   = ?
```



```
1 type AllUpper<T> = T extends Array<infer U> ? Uppercase<U> : never;
```



```
1 type ExtractChildren<T extends RouteConfig[], Base extends string = ''>
2   = T extends [infer Head extends RouteConfig, ...infer Tail extends RouteConfig[]]
```



```
1 type ExtractChildren<T extends RouteConfig[], Base extends string = ''>
2   = T extends [infer Head extends RouteConfig, ...infer Tail extends RouteConfig[]]
```



```
1 type ExtractChildren<T extends RouteConfig[], Base extends string = ''>
2   = T extends [infer Head extends RouteConfig, ...infer Tail extends RouteConfig[]]
```



```
1 type ExtractChildren<T extends RouteConfig[], Base extends string = ''>
2   = T extends [infer Head extends RouteConfig, ...infer Tail extends RouteConfig[]]
3     ? ExtractConfig<Head, Base> & ExtractChildren<Tail, Base>
4     : {}
```



```
1 type ExtractChildren<T extends RouteConfig[], Base extends string = ''>
2   = T extends [infer Head extends RouteConfig, ...infer Tail extends RouteConfig[]]
3     ? ExtractConfig<Head, Base> & ExtractChildren<Tail, Base>
4     : {}
```



```
1 type ExtractChildren<T extends RouteConfig[], Base extends string = ''>
2   = T extends [infer Head extends RouteConfig, ..., infer Tail extends RouteConfig[]]
3     ? ExtractConfig<Head, Base> & ExtractChildren<Tail, Base>
4     : {}
```



```
1 type ExtractChildren<T extends RouteConfig[], Base extends string = ''>
2   = T extends [infer Head extends RouteConfig, ...infer Tail extends RouteConfig[]]
3     ? ExtractConfig<Head, Base> & ExtractChildren<Tail, Base>
4     : {}
```





```
1 type ExtractConfig<T extends RouteConfig, Base extends string = ''> =
2   {[K in T['id']]: `${Base}/${T['url']}`}
3   & ExtractChildren<T['children'], `${Base}/${T['url']}`>
4
5 type ExtractChildren<T extends RouteConfig[], Base extends string = ''>
6   = T extends [infer Head extends RouteConfig, ...infer Tail extends RouteConfig[]]
7     ? ExtractConfig<Head, Base> & ExtractChildren<Tail, Base>
8     : {}
9
10 type Paths = ExtractConfig<typeof ROUTE_CONFIG>;
```

```
const paths = generateRouteMap(ROUTE_CONFIG)
```



```
generatePath(paths.resourceId, {})
```

 id

 user

```
type ExtractConfigT extends RouteConfig {  
  extend: template string  
}
```



```
1  type RouteConfig = {  
2    id: string;  
3    url: string;  
4    children?: RouteConfig[];  
5  }
```



```
1  type RouteConfig = {  
2    id: string;  
3    url: string;  
4    children?: RouteConfig[];  
5  }
```

# О чем поговорим?

Как прокачать работу с урлами

Как прокачать работу с конфигами

Какие еще есть кейсы

# О чем поговорим?

Как прокачать работу с урлами

Как прокачать работу с конфигами

Какие еще есть кейсы

# Где посмотреть еще примеры?

# Где посмотреть еще примеры?

# Где посмотреть еще примеры?

React/React-dom dts файлы

# Где посмотреть еще примеры?

React/React-dom dts файлы

Библиотеки с типами (utility-types, type-fest, etc)

# Где посмотреть еще примеры?

React/React-dom dts файлы

Библиотеки с типами (utility-types, type-fest, etc)

TypeChallenges

# Внимание!

type Awaited<T> = ?

Оставляйте ваш код лучшим,  
чем он был