24 - Omit e Pick

Objetivos da Aula

- Conhecer as tipos Omit e Pick e como eles podem ser úteis
- Mostrar exemplos de uso de Omit e Pick

Snippets

1. Em um arquivo typescript qualquer escreva o código abaixo que possui a interface completa da API do GitHub usada no último exercício:

```
interface GithubUserResponse { login: string id: number node_id: string avatar_url: string gravatar_id: string url: string html_url: string followers_url: string following_url: string gists_url: string starred_url: string subscriptions_url: string organizations_url: string repos_url: string events_url: string received_events_url: string type: string site_admin: boolean name: string company: string blog: string location: string email: string hireable: boolean bio: string twitter_username: string public_repos: number public_gists: number followers: number following: number created_at: string updated_at: string } interface GithubRepoResponse { id: number node_id: string name: string full_name: string private: boolean owner: { login: string id: number node_id: string avatar_url: string gravatar_id: string url: string html_url: string followers_url: string following_url: string gists_url: string starred_url: string subscriptions_url: string organizations_url: string repos_url: string events_url: string received_events_url: string type: string site_admin: boolean } html_url: string description: string fork: boolean url: string forks_url: string keys_url: string collaborators_url: string teams_url: string hooks_url: string issue_events_url: string events_url: string assignees_url: string branches_url: string tags_url: string blobs_url: string git_tags_url: string git_refs_url: string trees_url: string statuses_url: string languages_url: string stargazers_url: string contributors_url: string subscribers_url: string subscription_url: string commits_url: string git_commits_url: string comments_url: string issue_comment_url: string contents_url: string compare_url: string merges_url: string archive_url: string downloads_url: string issues_url: string pulls_url: string milestones_url: string notifications_url: string labels_url: string releases_url: string deployments_url: string created_at: string updated_at: string pushed_at: string git_url: string ssh_url: string clone_url: string svn_url: string homepage: string size: number stargazers_count: number watchers_count: number language: string has_issues: boolean has_projects: boolean has_downloads: boolean has_wiki: boolean has_pages: boolean forks_count: number mirror_url: string archived: boolean disabled: boolean open_issues_count: number license: string allow_forking: boolean is_template: boolean topics: string[] visibility: string forks: number open_issues: number watchers: number default_branch: string }
```

2. Bem grande, não é mesmo? Se repararmos bem vamos ver que bem ali na interface GithubRepoResponse nós temos uma propriedade *owner* que é um objeto muito semelhante ao da interface GithubUserResponse. Na verdade eles são iguais exceto por algumas poucas propriedades. E é aí que o **Omit** pode nos ajudar. Vejamos como utilizar o Omit substituindo o tipo de *owner* por:

```
owner: Omit<GithubUserResponse, 'name' | 'company' | 'blog' | 'location' | 'email' | 'hireable' | 'bio' | 'twitter_username' | 'public_repos' | 'public_gists' | 'followers' | 'following' | 'created_at' | 'updated_at' >
```

3. O Omit é um tipo genérico (veremos mais sobre eles nas próximas aulas) que permite criar um tipo a partir de outro , mas removendo algumas de suas propriedades. Ele recebe dois argumentos (utilizando a sintaxe < > característica dos genéricos), o primeiro é o tipo a ser copiado e o segundo são as propriedades a serem removidas na forma de uma união de strings. Se declararmos agora uma variável com o tipo GithubRepoResponse vemos através do autocompletar que ela não possui em owner os tipos que removemos:

```
let repo: GithubRepoResponse repo.owner.name = 'Isaac' // A propriedade 'name' não existe no tipo 'Omit<GithubUserResponse, ...
```

4. **O Omit é especialmente útil quando queremos utilizar quase todas propriedades de um tipo de objeto, mas queremos garantir que algumas não serão incluídas.** Outra forma de verificar seria criar um novo tipo RepositoryOnly que não possui a propriedade owner. Vemos que ela é completamente removida usando o Omit:

   Obs.: Como o Omit trabalha removendo propriedades mesmo que elas não existam ele não traz o autocompletar padrão do VS Code para o segundo argumento.

```
type RepositoryOnly = Omit<GithubRepoResponse, 'owner'> let repository: RepositoryOnly repository.owner = {} // A propriedade 'owner' não existe no tipo 'RepositoryOnly'
```

5. Semelhante ao Omit, o **Pick** possui a função contrária. Com ele é possível criar um tipo a partir outro, mas selecionando apenas algumas propriedades. Ele também um tipo genérico que recebe dois argumentos, o tipo a ser copiado e quais propriedades copiar na forma de uma união de strings. Vamos criar os tipos LocalGithubUser e LocalGithubRepository com as propriedades que utilizamos no último exercício:

   Obs.: Diferente do Omit, o Pick trabalha apenas dentro das possíveis propriedades do tipo a ser copiado, logo o autocompletar funciona até mesmo na hora de declarar a união de strings.

   Obs².: Como podemos ver, o caso de uso ideal do Pick é quando precisamos de apenas poucas propriedades de um tipo de objeto que já existe.

```
type LocalGithubUser = Pick<GithubUserResponse, 'id' | 'login' | 'name' | 'bio' | 'public_repos' | 'repos_url' > type LocalGithubRepository = Pick<GithubRepoResponse, 'name' | 'description' | 'fork' | 'stargazers_count' > let localUser: LocalGithubUser let localRepository: LocalGithubRepository // Vemos no autocompletar as propriedades de ambos os objetos localUser.name = '' localRepository.name = ''
```