Arrow Functions

Conteúdo

- sintaxe
- corpo conciso vs corpo de bloco
- arrow functions como métodos
- retornando objetos literais
- quebra de linha

Objetivo

Apresentar ao aluno a sintaxe de declaração de uma arrow function.

Sintaxe básica

A arrow function mais simples recebe um argumento e retorna um valor e pode ser declarada da maneira ao lado.

```
> let square = a => a * a;
<· undefined
> square(2)
<· 4
```

Não é preciso colocar o parâmetro entre parêntesis se houver só um.

Se a função não recebe nenhum parâmetro, declare-a com parênteses vazios.

```
> let foo = () => console.log("foo");
<· undefined
> foo();
  foo
```

Se a função precisa de mais de um parâmetro, declare-os entre parênteses.

```
> let multiply = (a, b) => a * b;
<· undefined
> multiply(2, 4);
<· 8
```

Corpo conciso VS Corpo de bloco

Uma arrow function pode ser declarada usando o corpo conciso. Se uma função tiver somente uma expressão, você pode declará-la sem colchetes e o retorno da função será o valor resultante da expressão.

```
> let multiply = (a, b) => a * b;
<· undefined
> multiply(2, 4);
<· 8
```

Também pode ter um corpo de bloco, onde você envolve as declarações entre colchetes e retornar um valor explicitamente com a palavra chave return.

```
> let multiply = (a, b) => {
      const result = a * b;

      return result;
  }
<· undefined
> multiply(2, 6);
<· 12
```

Arrow functions como métodos

Você pode usar uma arrow function para definir um método de um objeto.

```
> let obj = {
      attribute: "attribute",
      foo: () => {
          console.log("doing someting");

          return "something";
      }
  }
< undefined
> obj.foo();
  doing someting
< "something"
```

Retornando objetos literais

Se você quiser usar a sintaxe de corpo conciso para retornar um objeto literal, você deve envolvê-lo entre parênteses. De outro modo, o retorno não funcionará.

```
> let getObj = () => ({
      attr1: "atributo 1",
      attr2: "atributo 2",
      foo: () => {
          console.log("doing something");
      }
  });
```

Quebra de linha

Uma arrow function não pode conter uma quebra de linha entre os parâmetros e sua flecha.

```
> let foo = ()
      => 1;

Uncaught SyntaxError: Unexpected token '=>'
```

```
> let foo = (
      a,
      b
  ) => (
      1
  );
< undefined
> foo();
< 1
```

Ordem de análise

Apesar de a flecha numa arrow function não ser um operador, arrow functions possuem regras especiais de análise que interagem diferentemente com precedência de operador comparadas à funções comuns.

```
> let callback;
< undefined
> callback = callback || function() {};
< ƒ () {}
> callback = callback || () => {};
✕ Uncaught SyntaxError: Malformed arrow
  function parameter list
> callback = callback || (() => {});
< ƒ () {}
```

OBRIGADO!

WWW.ARBYTE.COM.BR