Arrow Function

ARROW FUNCTION

- Le arrow function sono funzioni che possono avere essere costituite in due modi.

  `(<IDENT>) => { <STAT>* }`

  oppure

  `(<IDENT>) => <EXPR>`

  nel caso ci sia solo un argomento è possibile non usare le parentesi

  `<IDENT> => ...`

ARROW FUNCTION

- Esempi

```
// Statement
(a, b) => {
console.log(a);
}
```

```
// Espressione
(a, b) => (a ? b : c)
(a, b) => a + b
```

ARROW FUNCTION

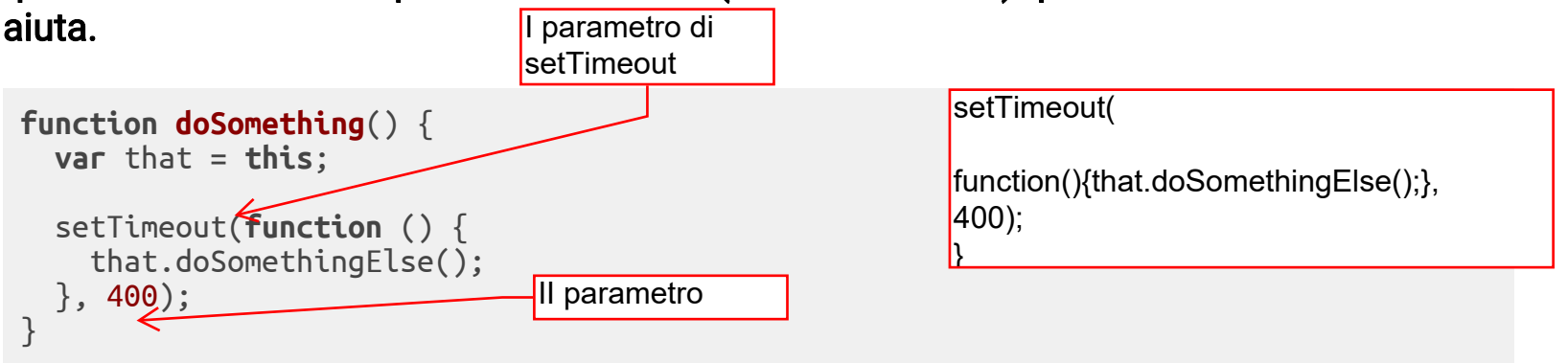
- ATTENZIONE
  - `() => {}` è un "Block**Statement**"
  - `() => ({})` è una "Object**Expression**"

```
[1,2,3,4].map(() => ({}));
// [{}, {}, {}, {}]
```

```
[1,2,3,4].map(() => {});
// [undefined, undefined, undefined, undefined]
```

ARROW FUNCTION

Le arrow function condividono lo stesso "lexical `this`" del codice che le circonda. **Nella quotidianità dello svillupo React moderno (ovvero via Hook) questa informazione non vi aiuta.**




```
function doSomething() {
  var that = this;

  setTimeout(function () {
    that.doSomethingElse();
  }, 400);
}
```

In ES2015 con le Arrow diventa:

```
function doSomething() {
  setTimeout(() => {
    this.doSomethingElse();
  }, 400);
}
```

ARROW FUNCTION

Le arrow condividono il parametro `arguments` della funzione all'interno della quale si trovano.

```
function myfunction() {
  let arrowfunction = () => {
    console.log(arguments);
  }
  arrowfunction();
}

myfunction(1, 2, 3);
// [1, 2, 3]
```

IIFE (IMMEDIATELY INVOKED FUNCTION EXPRESSION)

Una **Immediately Invoked Function Expression** è una *Espressione* costituita da una "CallExpression" che segue una "FunctionExpression"

- ATTENZIONE

```
function() {}()
// o
() => {}()
// Errore in compile time
```

```
(function() {}())
// o
(() => {})()
// FunctionExpression
```

In seguito a `function` l'interprete si aspetta una dichiarazione di una funzione con nome (Named Function Declaration). Le parentesi portano invece a intendere il contenuto come una espressione (all'interno di parentesi tonde si trova sempre una espressione come nell'esempio `{}`, `({})`)

IIFE (IMMEDIATELY INVOKED FUNCTION EXPRESSION)

- DOMANDA

```
let a = false;
let b = 1;
let c = 2;

((a, b) => a ? b : c)(a, b);
// ?
```

  - `true`
  - `false`
  - `1`
  - `2`
  - `TypeError`
  - Errore in fase di compilazione

IIFE (IMMEDIATELY INVOKED FUNCTION EXPRESSION)

- DOMANDA

```
let a = false;
let b = 1;
let c = 2;

((a, b) => a ? b : c)(a, b);
// ?
```

  - `true`
  - `false`
  - `1`
  - `2`
  - `TypeError`
  - Errore in fase di compilazione

RISPOSTA

  - 2

Function

Function

Diversi modi di instaziare una funzione

- Dichiarazione di funzione

```
function myfunc() {}
```

- Assegnazione di una FunctionExpression ad una variabile

```
let myfunc = function() {}
```

Function

Assegnazione di una FunctionExpression con nome ad una variabile

```
let myfunc = function myAwesomeName() {
  // Qui posso usare myAwesomeName!
}
```

In questo caso il nome `myAwesomeName` sarà raggiungibile solo all'interno della funzione stessa.

```
let myfunc = function myAwesomeName() {
  console.log(myAwesomeName);
};
myfunc();
// myAwesomeName() {...}
```

```
let myfunc = function myAwesomeName() {};
console.log(myAwesomeName);
// ReferenceError: myAwesomeName is not defined
```