ORIGAMID

MAIS JAVASCRIPT

Clojures e Debugging

Escopo

Quando criamos uma função, a mesma possui acesso à todas as variáveis criadas em seu escopo e também ao escopo pai. A mesma coisa acontece para funções dentro de funções.

```
let item1 = 1;
function funcao1() {
  let item2 = 2;
  function funcao2() {
    let item3 = 3;
  }
}

// func1, possui acesso à
// item1 e item2

// func2, possui acesso à
// item1, item2 e item3
```

Clojures

A `funcao2` possui 4 escopos. O primeiro escopo é o Local, com acesso ao item3. O segundo escopo dá acesso ao item2, esse escopo é chamado de Clojure (funcao1) (escopo de função dentro de função). O terceiro escopo é o Script com acesso ao item1 e o quarto escopo é o Global/Window.

```
let item1 = 1;
function funcao1() {
  let item2 = 2;
  function funcao2() {
    let item3 = 3;
    console.log(item1);
    console.log(item2);
    console.log(item3);
  }
  funcao2();
}
```

Debugging

É possível "debugarmos" um código JavaScript utilizando ferramentas do browser ou através do próprio Visual Studio Code. Se o código possuir qualquer Web API, o processo deve ser feito no Browser. Plugins podem interferir no debug dentro do browser.

```
debugger; // adicione a palavra debugger
let item1 = 1;
function funcao1() {
  let item2 = 2;
  function funcao2() {
    let item3 = 3;
    console.log(item1);
    console.log(item2);
    console.log(item3);
  }
  funcao2();
}
```

Caso Clássico

Um dos casos mais clássicos para a demonstração de Clojures é através da criação de uma função de incremento. É como se a função incrementar carregasse uma mochila chamada contagem, onde uma referência para as suas variáveis estão contidas na mesma.

```
function contagem() {
  let total = 0;
  return function incrementar() {
    total++;
    console.log(total);
  }
}

const ativarIncrementar = contagem();
ativarIncrementar(); // 1
ativarIncrementar(); // 2
ativarIncrementar(); // 3
```

Clojures na Real

Todas as funções internas da Factory Function possuem uma closure de `$$`. As mesmas contém uma referência à variável elements declarada dentro do escopo da função.

```
function $$(selectedElements) {
  const elements = document.querySelectorAll(selectedElements);

  function hide() { ... }
  function show() { ... }
  function on() { ... }
  function addClass() { ... }
  function removeClass() { ... }

  return { hide, show, on, addClass, removeClass }
}
```