Optimización funciones recursivas

IIC3585

Andres Matte Vallejos

```
// Sumatoria con recursion normal
const sum = (n) => {
  if (n <= 1) return n;
  else return n + sum(n-1);
};
```

```
sum(16000);
```

Uncaught RangeError: Maximum call stack size exceeded

¿Qué está pasando?

```
// Sumatoria con recursion normal
const sum = (n) => {
  if (n <= 1) return n;
  else return n + sum(n-1);
};
```

- Se añade un *Stack Frame* al Stack cada vez que se hace el llamado a la función recursiva.

- 16.000 Stack Frames son suficientes para sobrepasar el límite de tamaño del Stack.

¿Cómo lo solucionamos?

- Usar Proper Tail Call (Optimization).
  - Funcionalidad en ES6.

- Usar una función `trampoline`.

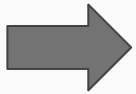
Proper Tail Call (PTC)

- Funciones que cumplen con ciertas condiciones son optimizadas para ocupar siempre el mismo Stack Frame.
- Ya no se agrega un Stack Frame por cada llamado a la función recursiva.
- Funcionalidad en ES6.
- Hay que transformar la función a una Tail Recursive.

Proper Tail Call (PTC)

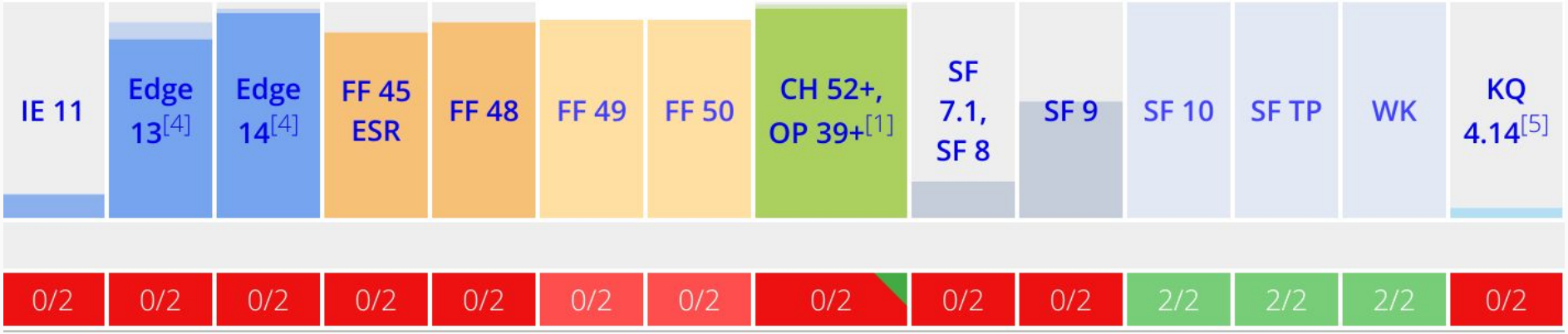
Transformar a forma Tail Recursive.

```
// Sumatoria con recursion normal
const sum = (n) => {
  if (n <= 1) return n;
  else return n + sum(n-1);
};
```



```
// Sumatoria en forma Tail Recursive
const sum = (n, acc) => {
 if (n <= 1) return acc;
 else return sum(n-1, acc + n);
};
```

Proper Tail Call (PTC)



Proper Tail Call (PTC)

Prueba en Safari 10?

Trampoline

- Técnica para mantener solo una operación a la vez en el Stack.
- Se delega la llamada recursiva a una función iterativa: `trampoline`.
- Es más lento que llamada recursiva normal, pero no hay problemas con el Stack.

Trampoline

```
const trampoline = (fn) => {
  while (typeof fn === 'function') {
    fn = fn();
  }
  return fn;
};
```

```
const sum = (n, acc) => () => {
  if (n <= 1) return acc;
  else return sum(n-1, acc + n);
};
```

```
// No hay problemas! :D
console.log(trampoline(sum(160000, 0)));
```

Referencias

- https://webkit.org/blog/6240/ecmascript-6-proper-tail-calls-in-webkit/
- http://eddmann.com/posts/recursive-functions-using-a-trampoline-in-javascript/
- https://www.sitepoint.com/recursion-functional-javascript/
- https://taylodl.wordpress.com/2013/06/07/functional-javascript-tail-call-optimization-and-trampolines/