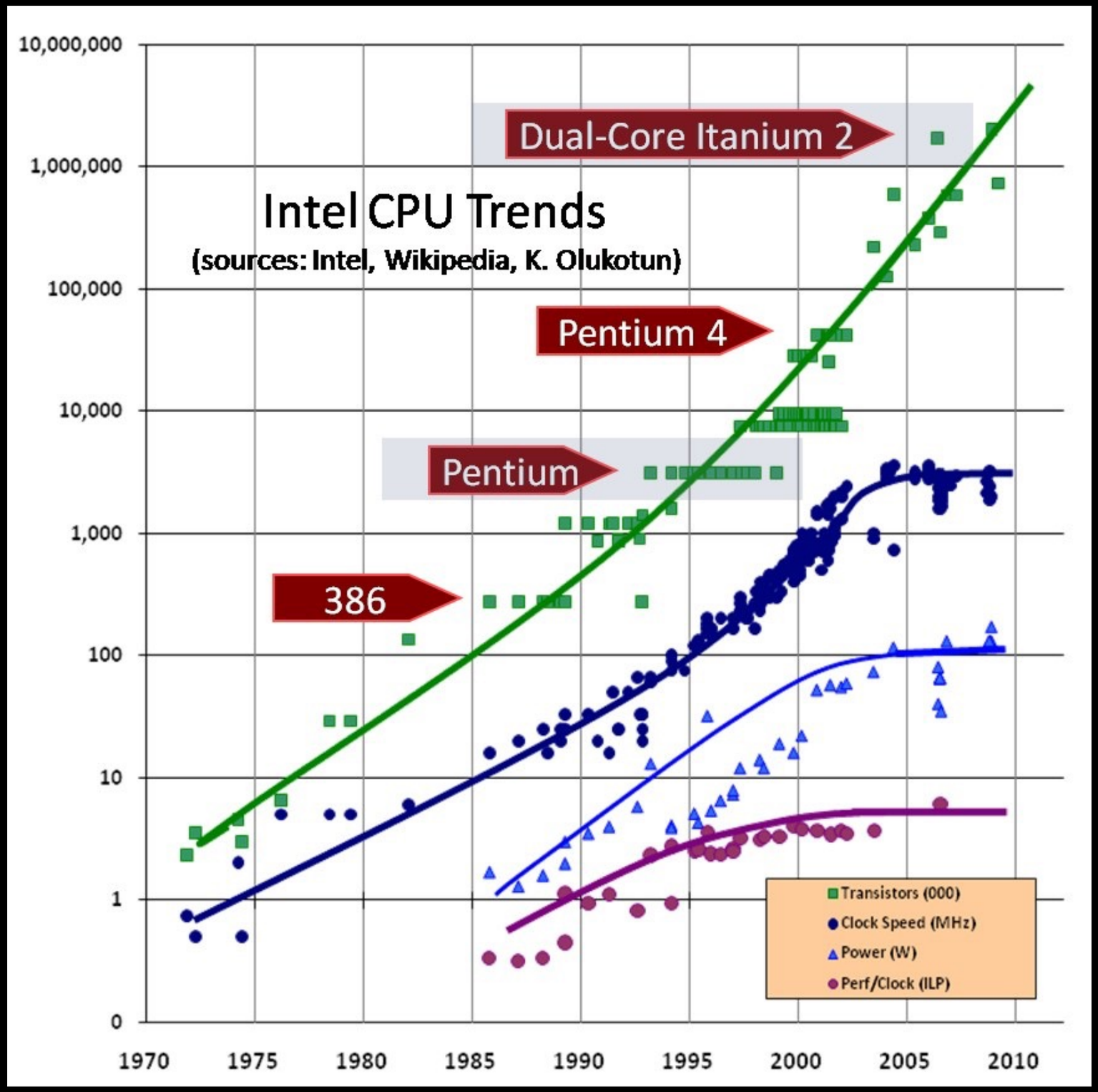
Utilización de esqueletos algorítmicos en EcmaScript para paralelizar el computo en navegadores Web a través de Web Workers

Schenkelman, Damian
Servetto, Matias

Tutora: Rosa Wachenchauzer



http://www.extremetech.com/wp-content/uploads/2013/08/CPU-Scaling.jpg

Paralelización

Comunicación de datos

Comunicación de código

Sincronización

Consolidación de resultados

Esqueletos algorítmicos

Modelo de alto nivel para programación paralela

Operaciones trivialmente paralelizables

```
doubles = map(e => e * 2, [1,2,3,4]);

// doubles [2,4,6,8]
```

map

map :: (a -> b) -> [a] -> [b]

```
dobles = map(e => e * 2, [1,2,3,4]);

// dobles [2,4,6,8]
```

map

map :: (a -> b) -> [a] -> [b]

```
pares = filter(e => e % 2 == 0, [1,2,3,4]);

// pares [2,4]
```

filter

filter :: (a -> Bool) -> [a] -> [a]

```
pares = filter(e => e % 2 == 0, [1,2,3,4]);

// pares [2,4]
```

filter

filter :: (a -> Bool) -> [a] -> [a]

```
suma = reduce((ac, cur) => ac + cur,
  0,
  [1,2,3,4]);

// suma 10
```

reduce

reduce :: (a -> b -> a) -> a -> [b] -> a

```
suma = reduce((ac, cur) => ac + cur,
  0,
  [1,2,3,4]);

// suma 10
```

reduce

reduce :: (a -> b -> a) -> a -> [b] -> a

```
pipe(filter(e => e % 2 == 0),
  map(e => e * 2), [1,2,3,4]);

// [4, 8]
```

Encadenamiento

pipe :: ([a] -> [b]) -> ([b] -> [c]) -> [a] -> [c]

```
pipe(filter(e => e % 2 == 0),
  map(e => e * 2), [1,2,3,4]);

// [4, 8]
```

Encadenamiento

pipe :: ([a] -> [b]) -> ([b] -> [c]) -> [a] -> [c]

Caso de estudio

UI

[1,2,3,4]

e => e * 2

T1

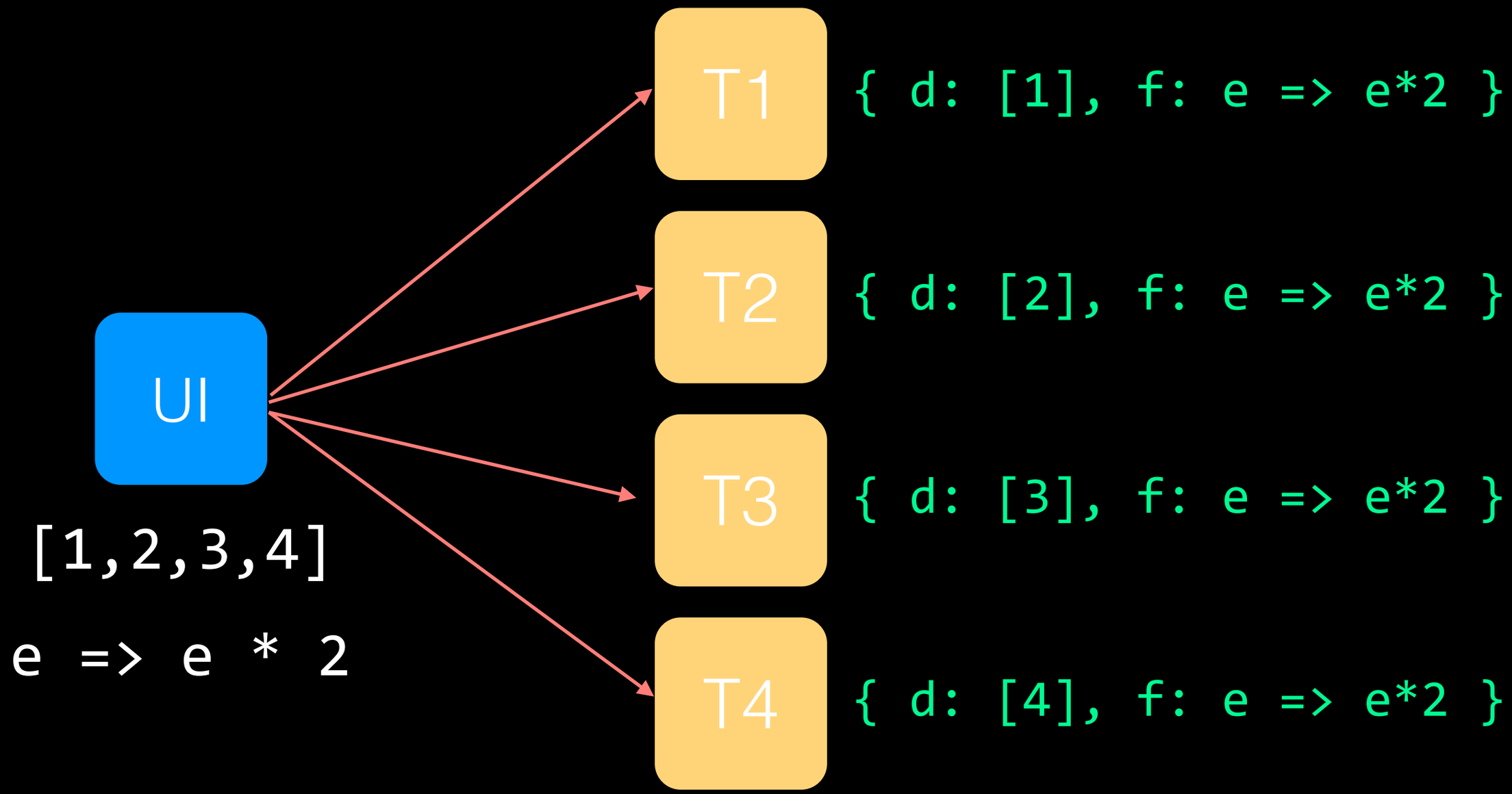
T2

T3

T4

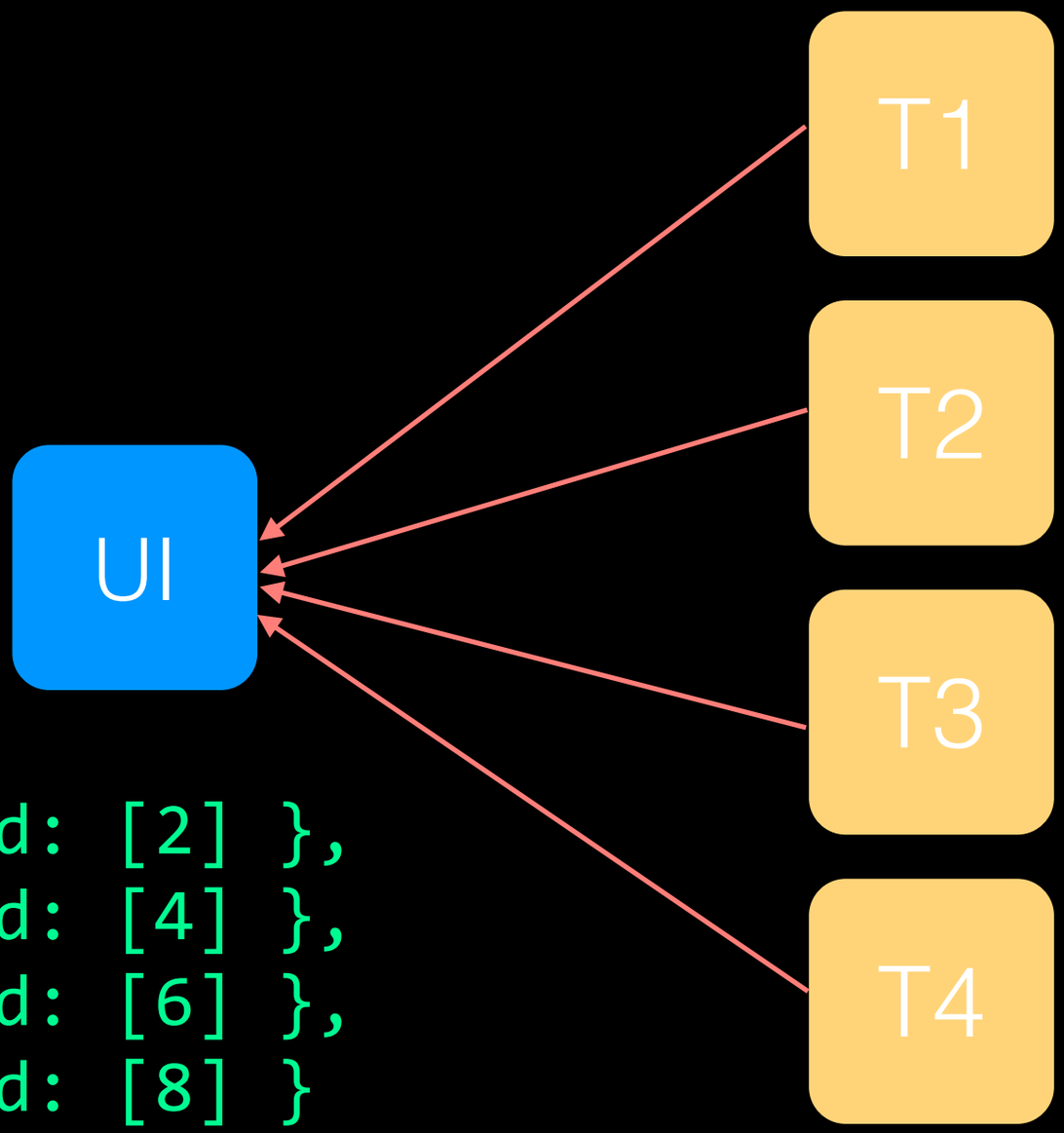
map

Ejemplo



map

Ejemplo (2)




```
[
  { d: [2] },
  { d: [4] },
  { d: [6] },
  { d: [8] }
]
```

map

Ejemplo (3)

UI

[2,4,6,8]

T1

T2

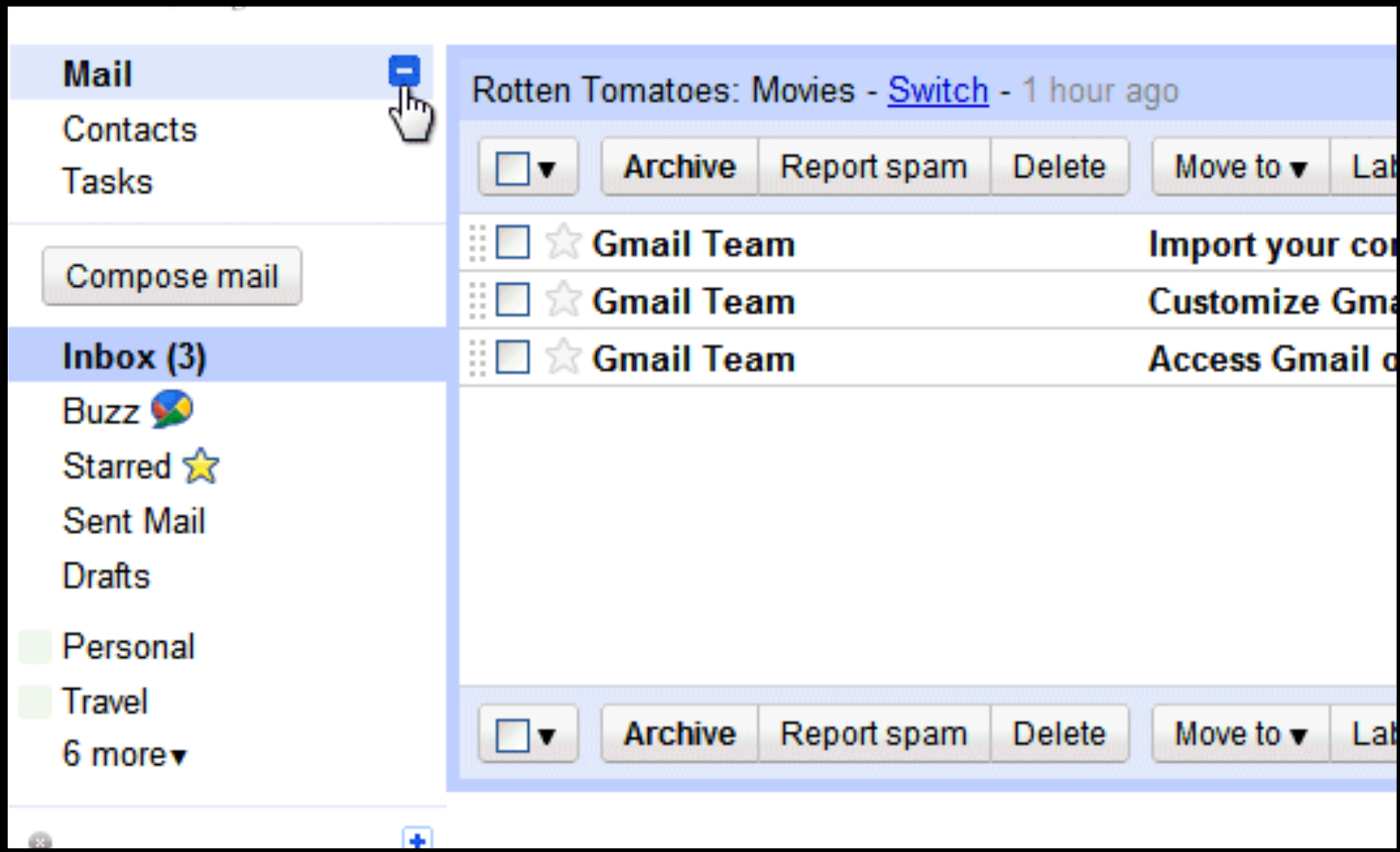
T3

T4

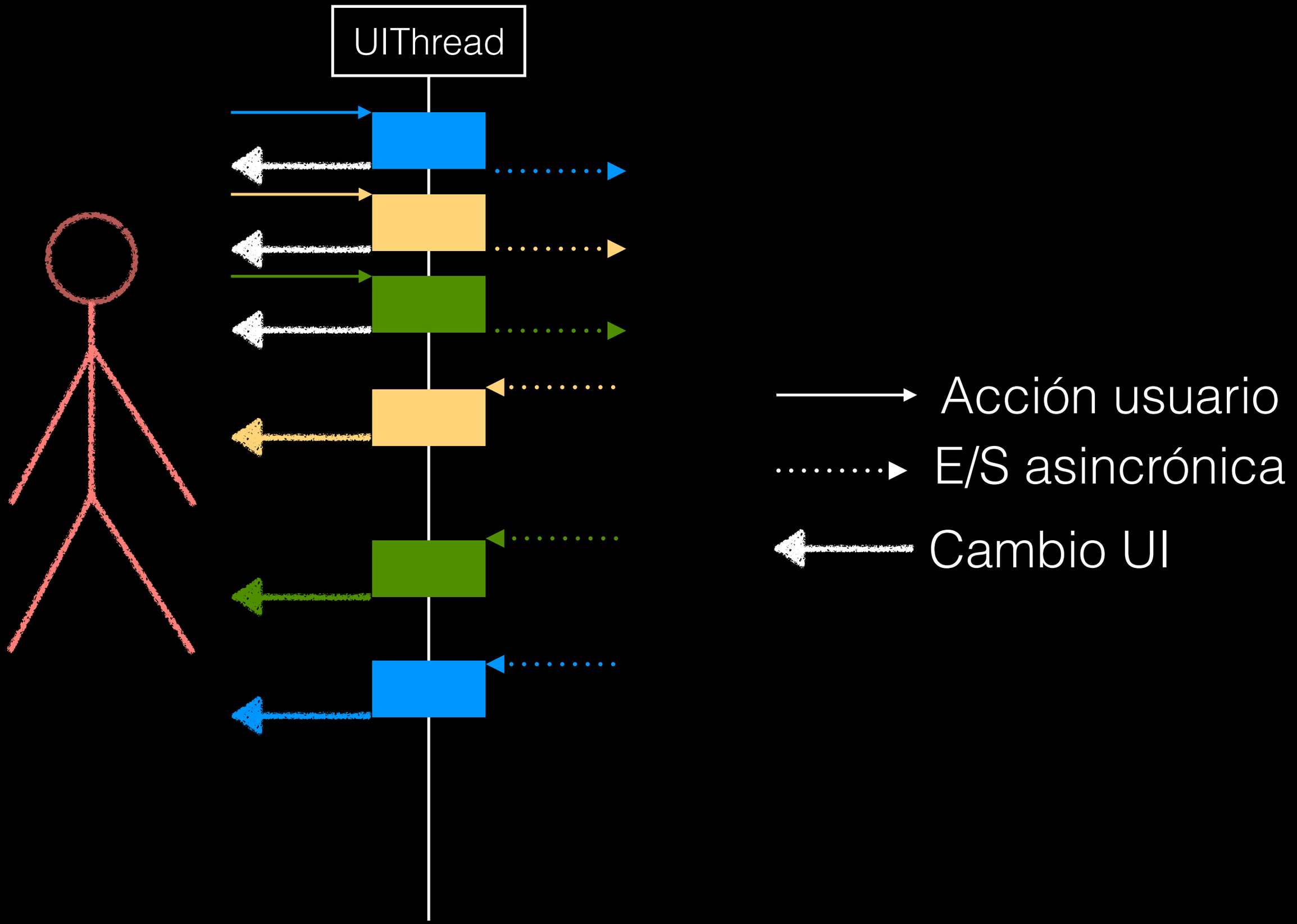
map

Ejemplo (4)

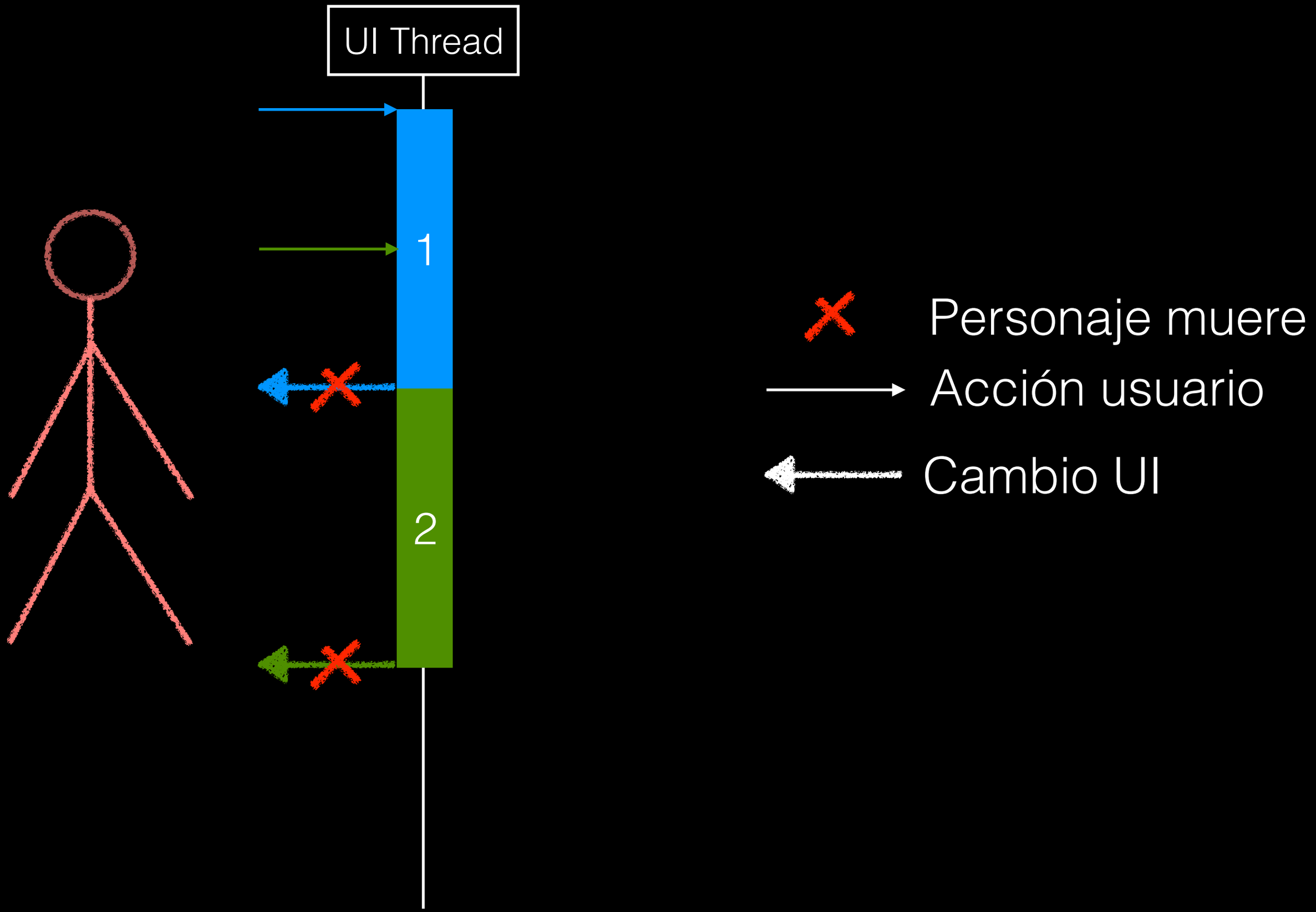
JavaScript

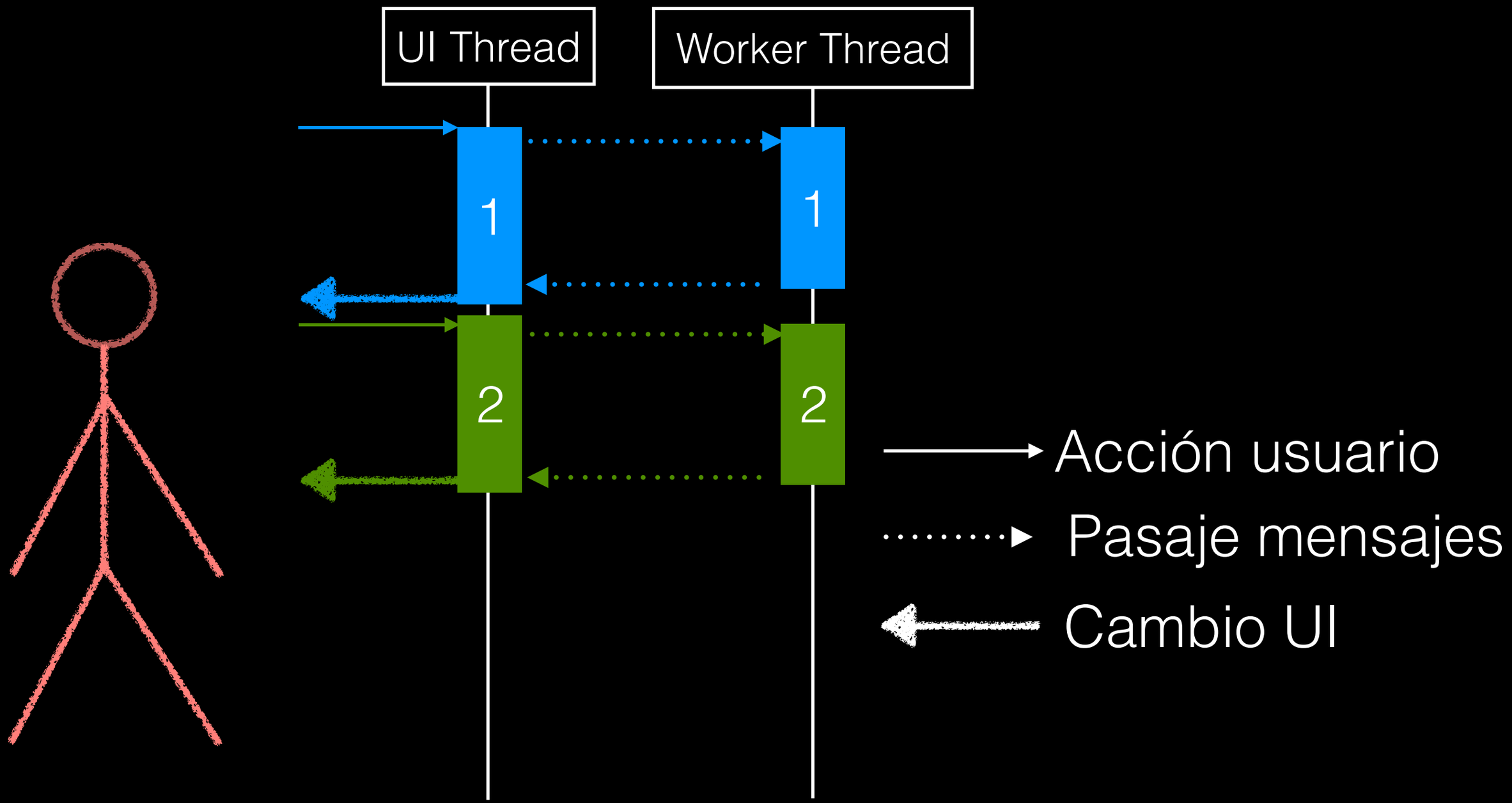

Mail
Contacts
Tasks
Compose mail
Inbox (3)
Buzz
Starred
Sent Mail
Drafts
Personal
Travel
6 more
Rotten Tomatoes: Movies - Switch - 1 hour ago
Archive
Report spam
Delete
Move to
Gmail Team
Import your co
Gmail Team
Customize Gma
Gmail Team
Access Gmail c
Archive
Report spam
Delete
Move to

GMail (2010)


UIThread
Acción usuario
E/S asincrónica
Cambio UI

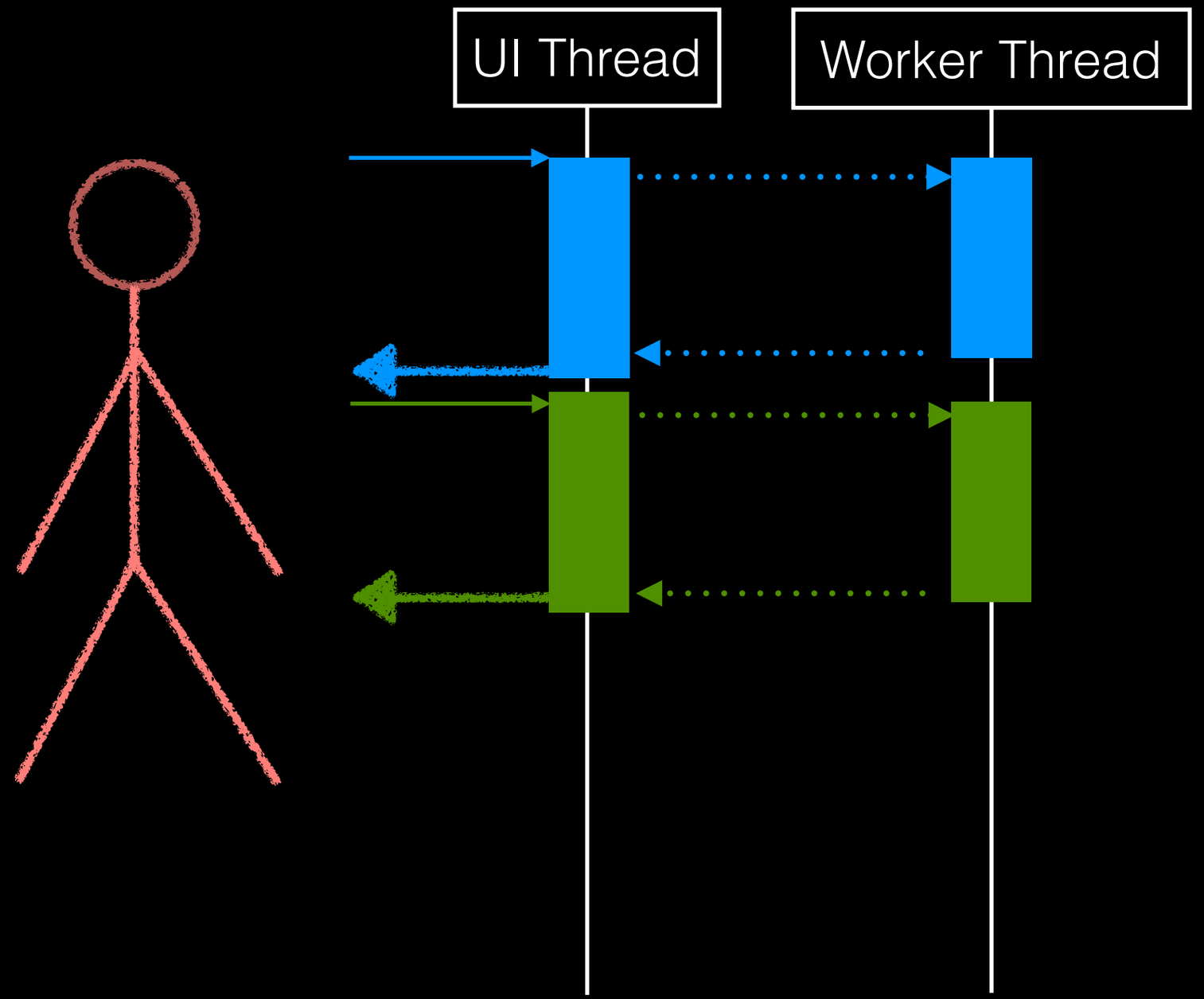
Unreal Engine en Firefox con asm.js


UI Thread
1
2
Personaje muere
Acción usuario
Cambio UI



Dividir el trabajo

Paralelización



Web Workers

index.html

```
const w = new Worker('worker.js');
w.addEventListener('message', function(e){
  console.log('Recibido del worker', e.data);
  worker.terminate();
}
worker.postMessage('Hola mundo');
```

worker.js

```
self.addEventListener('message', function(e) {
  const data = e.data;
  self.postMessage(data + '!!!');
});
```

Uso básico

Soportado

- XHR
- `importScripts(scriptName)`
- Crear otros workers

No soportado

- DOM
- `window`
- `document`
- `parent`

Sintaxis

```
var xs = [1,2,3,4];

var dobles = xs.map(x => x * 2);

var pares = xs.filter(x => x % 2 == 0);

var suma = xs.reduce((ac, el) => ac + el, 0);
```

JavaScript

map, reduce, filter

```
var pjs = require('p-j-s');

// cantidad de workers para el pool
pjs.init({ maxWorkers: 4 });

var xs = new Uint32Array([1,2,3,4]);

pjs(xs).map(e => e * 2).seq()
// asincrónico
.then(cuadrados => {
  // res [2,4,6,8]
});
```

p-j-s

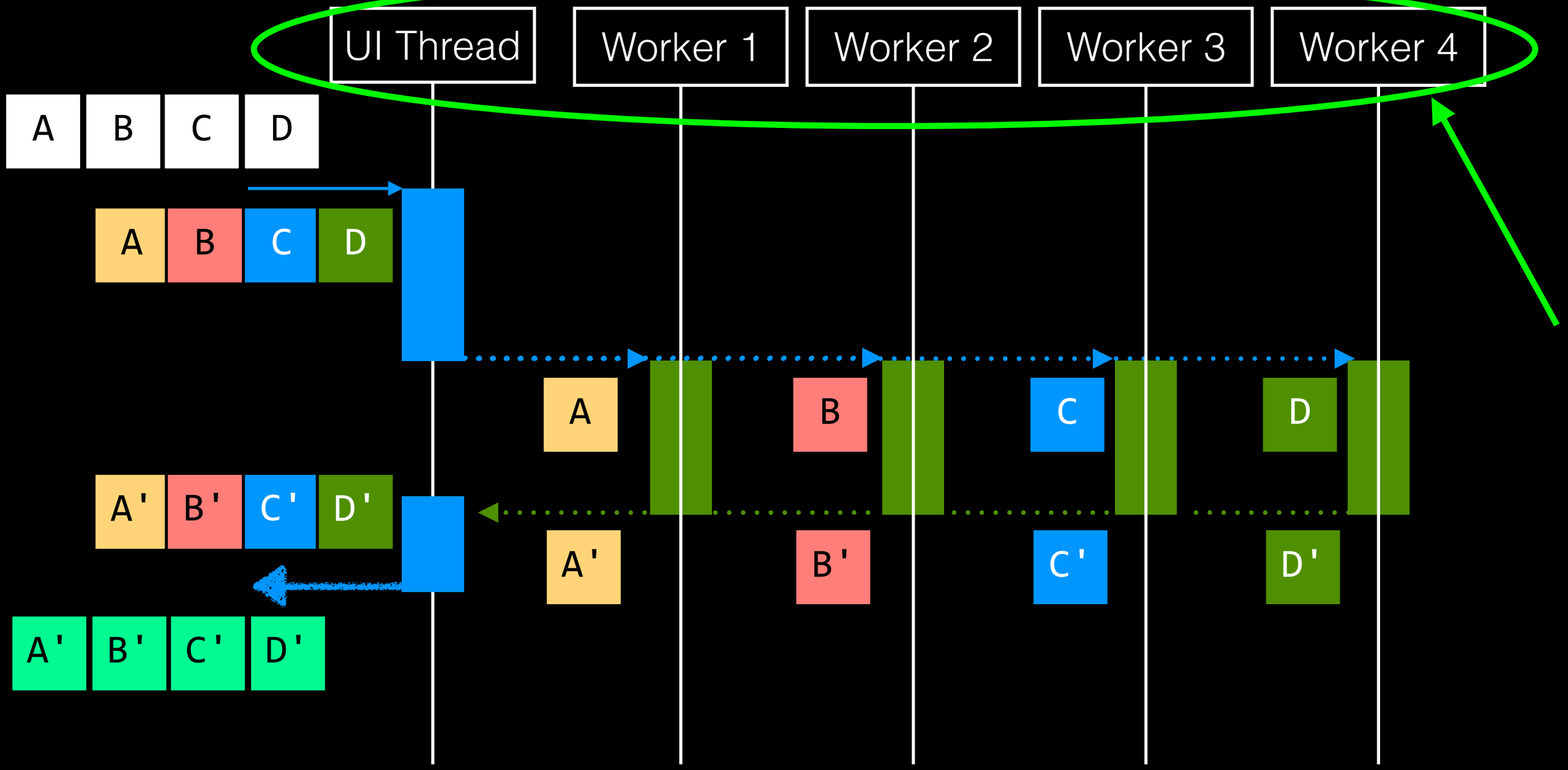
Básico

```
var xs = [1,2,3,4];

var ys = xs
 // paso 1
 .filter(x => x % 2 == 0)
 // paso 2
 .map(x => x * 2);

// ys [4,8]
```

```
var xs = new Uint32Array([1,2,3,4]);

pjs(xs)
  // paso 1
  .filter(x => x % 2 == 0)
  // paso 2
  .map(x => x * 2)
  // terminar cadena
  .seq().then(ys => {
   // ys [4,8]
  });
```

Comparación

Inicialización


UI Thread
Worker 1
Worker 2
Worker 3
Worker 4
A
B
C
D
A
B
C
D
A
B
C
D
A'
B'
C'
D'
A'
B'
C'
D'

p-j-s

Navegador	Worker Pool	Cold start
Chrome 40	6.213 ops/sec	120 ops/sec (98% más lento)
FF Nightly 41	4.622 ops/sec	152 ops/sec (96% más lento)

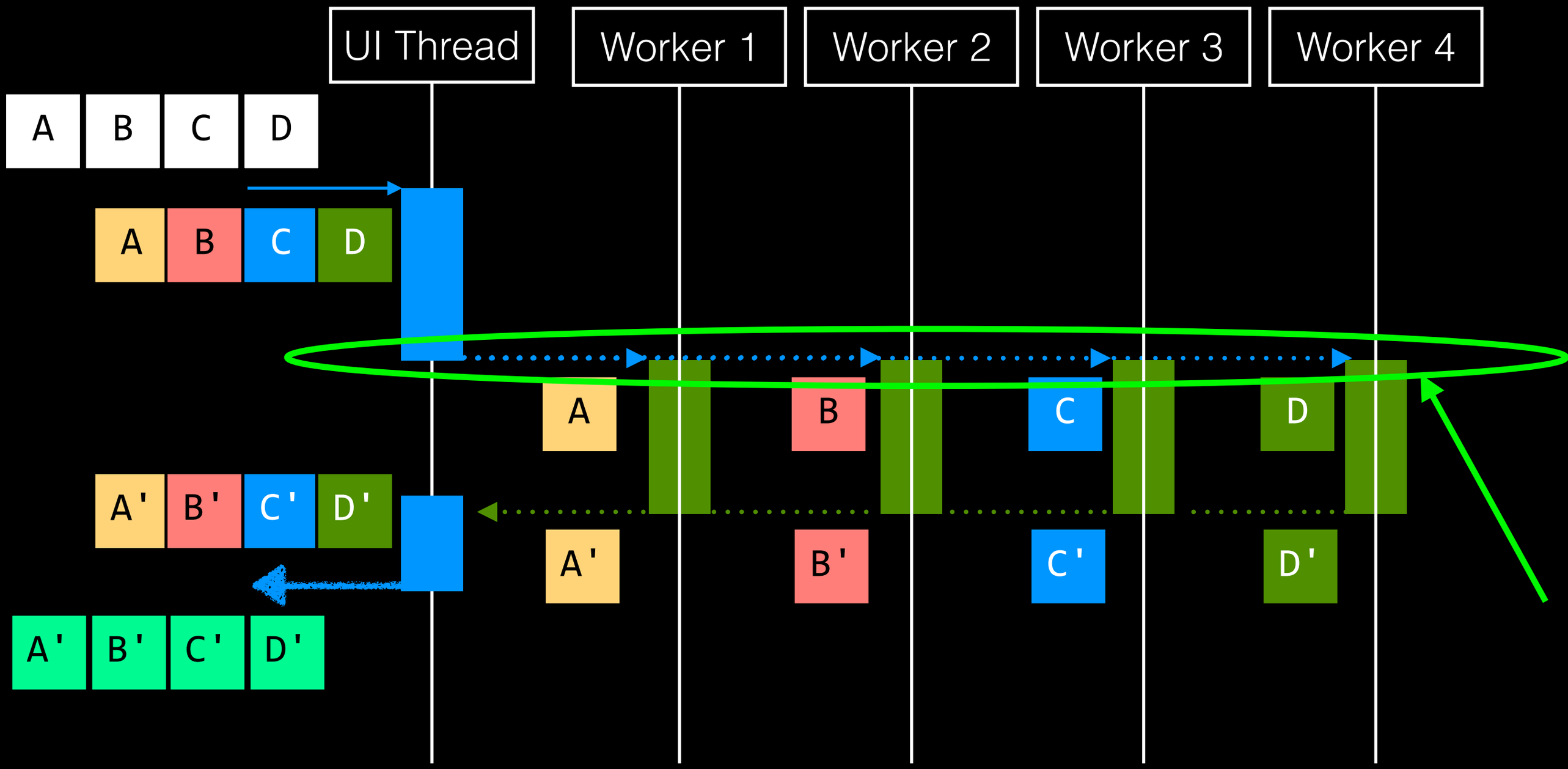
Creación de Workers

http://jsperf.com/worker-cold-start/6

Cantidad de Workers

- Memoria consumida por instancia
- "Intercalación" (*interleaving*)

Comunicación de datos


UI Thread
Worker 1
Worker 2
Worker 3
Worker 4
A
B
C
D
A
B
C
D
A
B
C
D
A'
B'
C'
D'
A'
B'
C'
D'
A'
B'
C'
D'

p-j-s

$$T_{ser} = \sum_{i=0}^{N} t = Nt$$

$T_{ser} : Tiempo\ en\ serie$

$N : Cantidad\ de\ elementos$

$t : Tiempo\ promedio\ por\ elemento$

Tiempo en serie

$$T_{par} = \frac{T_{ser}}{K}$$

$T_{ser} : Tiempo\ en\ serie$

$T_{par} : Tiempo\ en\ paralelo$

$K : Cantidad\ de\ Workers$

Tiempo en paralelo

Ideal

$$T_{sync} = T_t + T_s + T_p$$

$$T_{par} \approx \frac{T_{ser}}{K} + T_{sync}$$

$T_t : Tiempo\ de\ transferencia$

$T_s : Tiempo\ de\ serializacion/deserializacion$

$T_p : Tiempo\ de\ particion/union$

Tiempo en paralelo

Real

spec

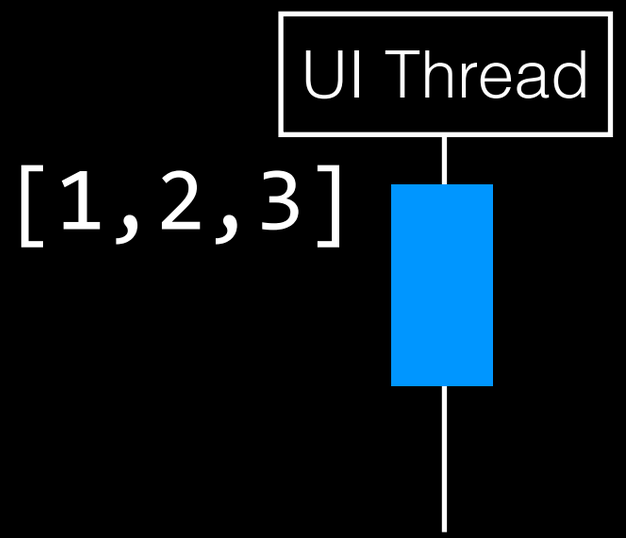
```
void postMessage(any message ,
optional sequence < Transferable > transfer)
```

ejemplo

```
const worker = new Worker('worker.js');
...
const items = new Uint8Array([1,2,3,4]);
worker.postMessage(items, [ items.buffer ]);
```

postMessage

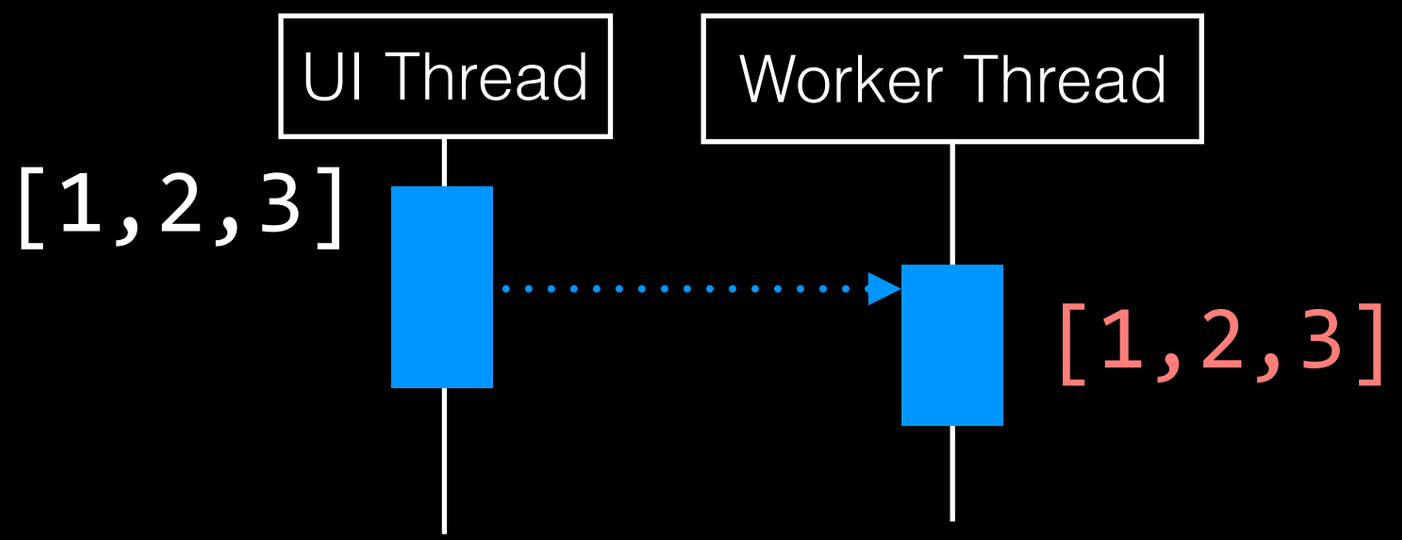
```
const xs = new Uint32Array([1,2,3]);
```



Cloning

No transferable

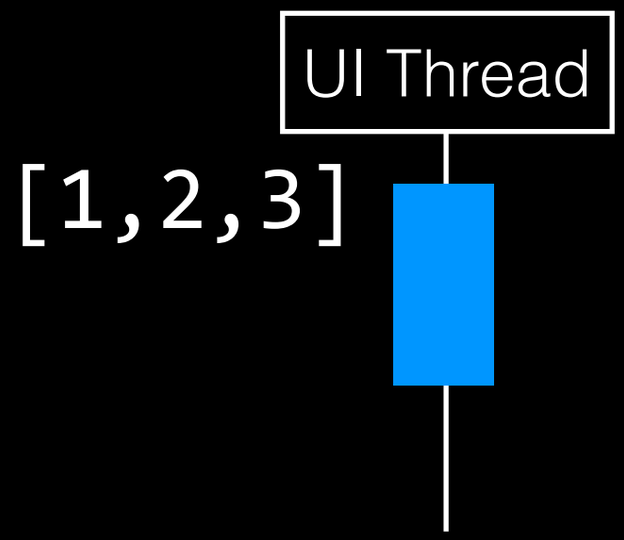
```
const xs = new Uint32Array([1,2,3]);
worker.postMessage(xs);
```



Cloning (2)

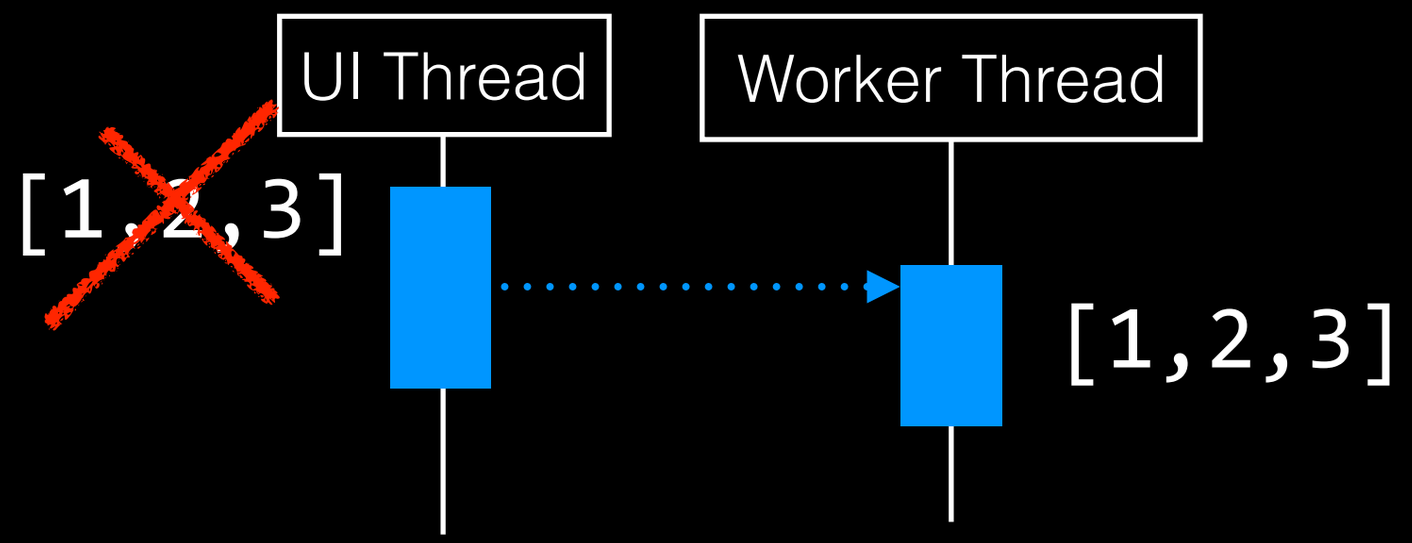
No transferable

```
const xs = new Uint32Array([1,2,3]);
```



Transferables

```
const xs = new Uint32Array([1,2,3]);
worker.postMessage(xs, [ xs.buffer ]);
```



Transferables(2)

Navegador	Transferable	Cloning
Chrome 40 (100,000 items)	8.793 ops/sec	1.513 ops/sec (82% más lento)
Chrome 40 (1.000.000 items)	8.767 ops/sec	58 ops/sec (99% más lento)

TypedArrays

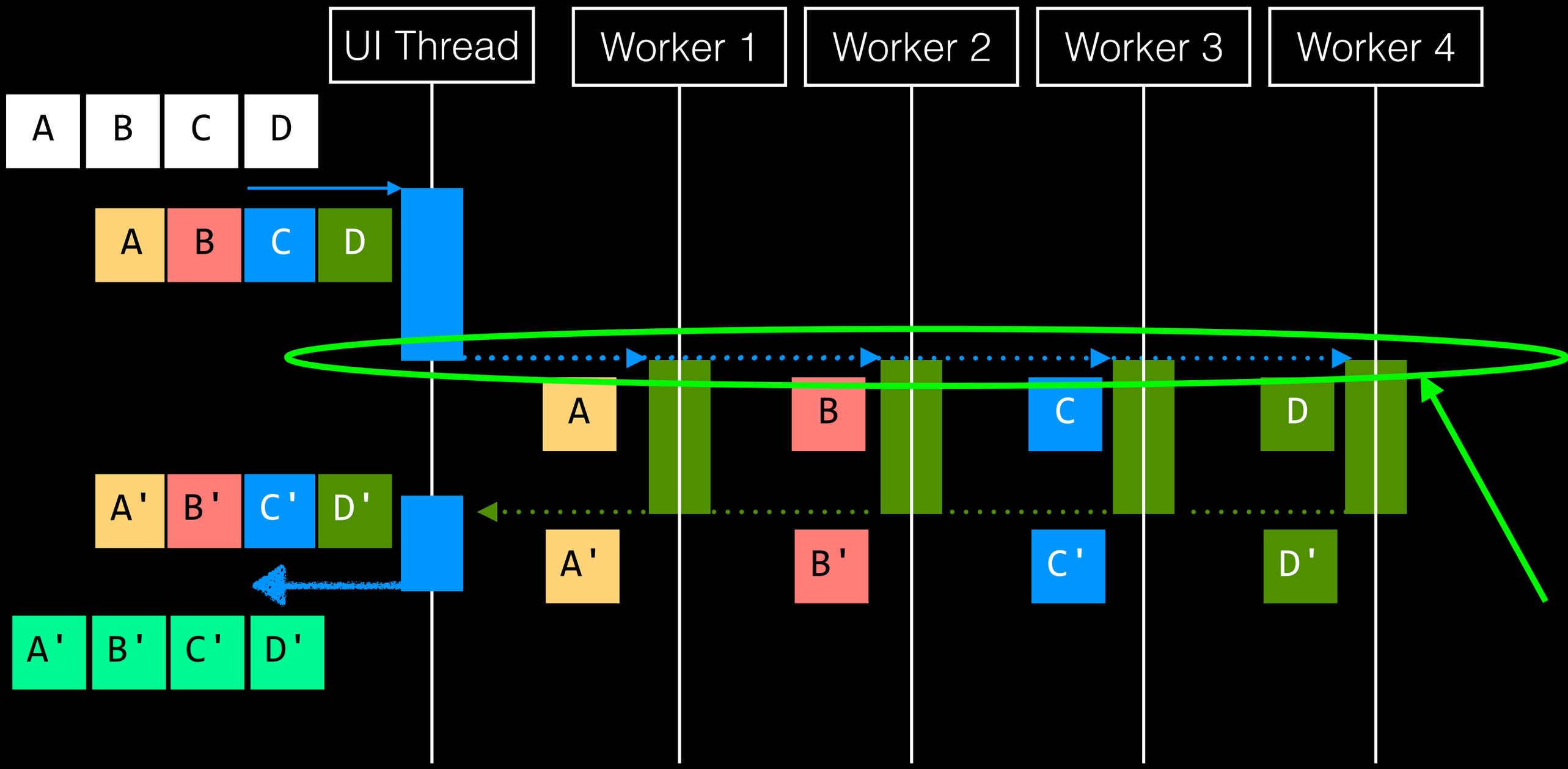
http://jsperf.com/transferrable-vs-cloning/3
http://jsperf.com/longer-transferrable-vs-cloning/3

Navegador	SharedTypedArray	TypedArray
FF Nightly 41 (100.000 items)	6.223 ops/sec	435 ops/sec (93% más lento)

SharedTypedArrays

http://jsperf.com/transfer-shared-vs-not-shared

Comunicación de código


UI Thread
Worker 1
Worker 2
Worker 3
Worker 4
A
B
C
D
A
B
C
D
A
B
C
D
A'
B'
C'
D'
A'
B'
C'
D'
A'
B'
C'
D'

p-j-s

```
> var f = function(a) { return a + 1; };
> worker.postMessage(f);
Uncaught DOMException: Failed to execute 'postMessage'
on 'Worker': An object could not be cloned.
```

Enviar funciones

```
> var f = new Function([ 'a' ], 'return a + 1’);
> f(2)
3
```

Alternativa

Navegador	Nativo	No nativo
Chrome 40	12.811 ops/sec	11.901 ops/sec (7% más lento)
FF Nightly 41	5.139 ops/sec	4.685 ops/sec (9% más lento)

Codificación

Para utilizar con ArrayBuffer

http://jsperf.com/pjs-serialization-comparison/4

Navegador	Copia	Transferable
Chrome 40	11.044 ops/sec	7.107 ops/sec (36% más lento)
FF Nightly 41	5.545 ops/sec	4.845 ops/sec (13% más lento)

Pasaje mensaje

Corto, sin memoria compartida
http://jsperf.com/pjs-serialization/5

Navegador	Copia	Transferable
Chrome 40	10.736 ops/sec	7.106 ops/sec (34% más lento)
FF Nightly 41	3.273 ops/sec	2.967 ops/sec (9% más lento)

Pasaje mensaje

Largo, sin memoria compartida

http://jsperf.com/pjs-serialization-long/3

Navegador	Copia	Transferable
FF Nightly 41	1.315 ops/sec	1.226 ops/sec (7% más lento)

Pasaje mensaje

Con memoria compartida, a cuatro Web Workers

http://jsperf.com/shared-function-transfer

Contexto

Serialización de clausuras

```
var xs = [1,2,3,4];
var fact = 4;

var ys = xs.map(function(x){
  return x * fact;
});
// ys [ 4, 8, 12, 16 ]
```

Clausuras

Closures

Contexto Global

El mismo para todas las cadenas

Contexto local

Específico para un paso de una cadena

```
var xs = new Uint8Array([1,2,3,4]);

// contexto global
pjs.updateContext({
  max: 3
}).then(() => {
  pjs(xs).filter(
    (e, ctx) => e <= ctx.max && e >= ctx.min,
    /* contexto local */ { min: 2 }
  )
  .seq().then(ys => {
    // ys [2 ,3]
  });
});
```

Contexto

Ejemplo (1)

```
pjs.updateContext({ A: 0, B: 25, C: 44 })
  .then(() => {
    var chain = pjs(xs)
      .map(
        (e, ctx) => { /* code here */ },
        { A: 5, B: 13 }
      ).filter(
        (e, ctx) => { /* code here */ },
        { A: 2, E: 9 }
      );

    ...
  }
});
```

Contexto

Ejemplo (2)

Local Context map	
A	5
B	13

Global Context	
A	0
B	25
C	44

Available ctx map	
A	5
B	13
C	44

Local Context filter	
A	2
E	9

Global Context	
A	0
B	25
C	44

Available ctx filter	
A	2
B	25
C	44
E	9

Union de contextos

Navegador	Mismo mensaje	Mensajes distintos
Chrome 40	6.089 ops/sec	5.497 ops/sec (10% más lento)
FF Nightly 41	389 ops/sec (2% más lento)	396 ops/sec

Pasaje contexto

Enviar en mismo mensaje que código o separado
http://jsperf.com/additional-sendingdata-comparison/2

Navegador	JSON.stringify	Cloning
Chrome 40	6.576 ops/sec	6.268 ops/sec (5% más lento)
FF Nightly 41	435 ops/sec	403 ops/sec (7% más lento)

Serialización

Contexto pequeño

http://jsperf.com/additional-clonningdata-comparison/3

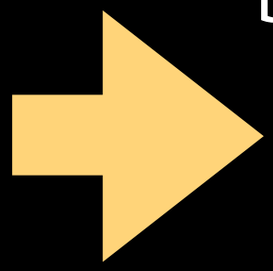
Navegador	JSON.stringify	Cloning
Chrome 40	6.309 ops/sec	5.317 ops/sec (16% más lento)
FF Nightly 41	405 ops/sec (3% más lento)	416 ops/sec

Serialización

Contexto grande

http://jsperf.com/additional-clonningdata-comparison-long/3

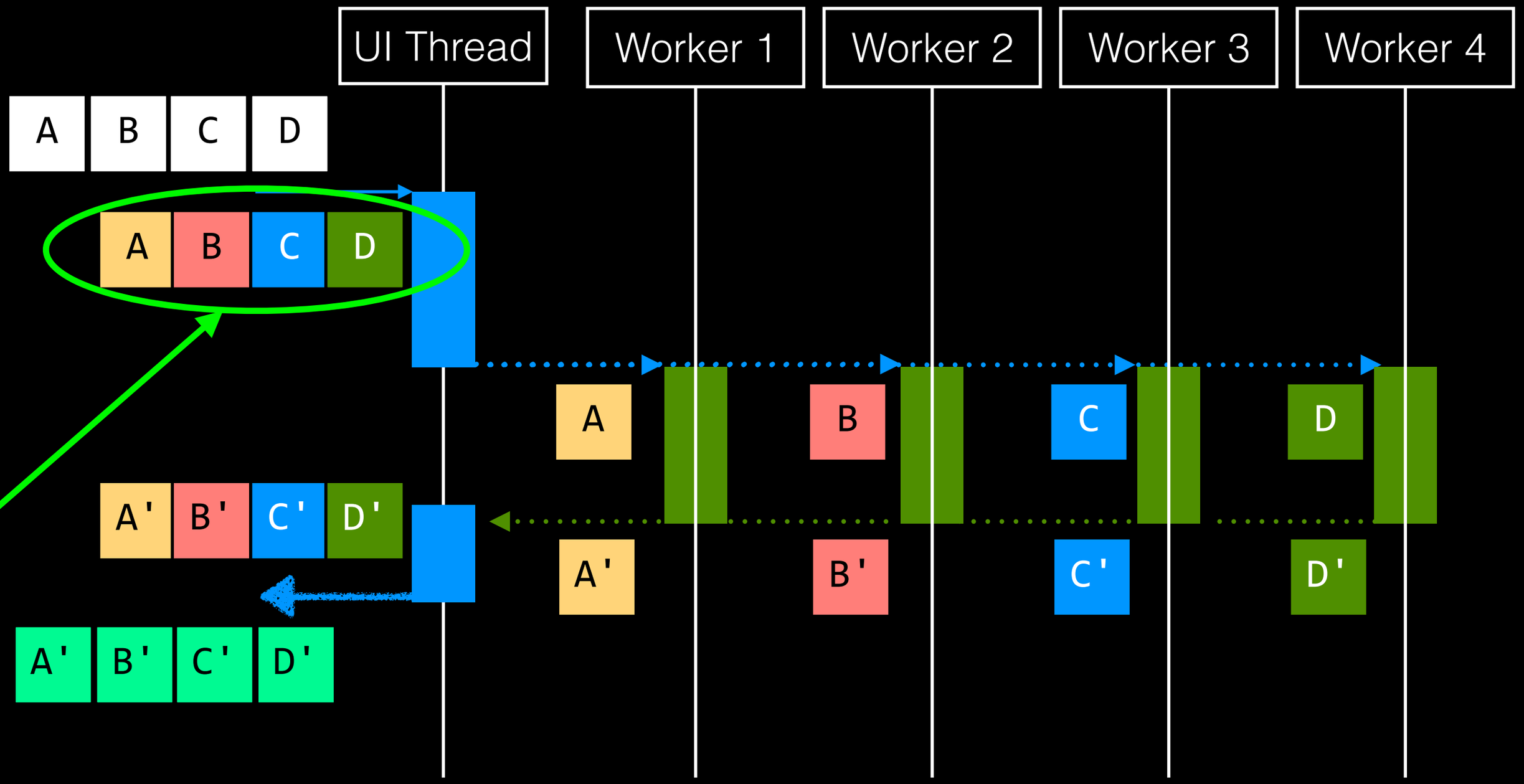
```
{
  a: 2,
  b: function(c) {
    return c + 1;
  }
}
```



```
{
  a: 2,
  b: {
    args: [ 'c' ],
    code: '\n\treturn c + 1;\n',
    __isFunction: true
  }
}
```

Contexto con funciones

Partición de TypedArray(s)


UI Thread
Worker 1
Worker 2
Worker 3
Worker 4
A
B
C
D
A
B
C
D
A
B
C
D
A'
B'
C'
D'
A'
B'
C'
D'
A'
B'
C'
D'

p-j-s

Navegador	Subarray + Constructor	Subarray + Set	Buffer slice + Constructor
Chrome 40	2868 ops/sec	2.234 ops/sec (22% más lento)	2.247 ops/sec (22% más lento)
FF Nightly 41	1.613 ops/sec (5% más lento)	1.638 ops/sec (3% más lento)	1.692 ops/sec

Partición

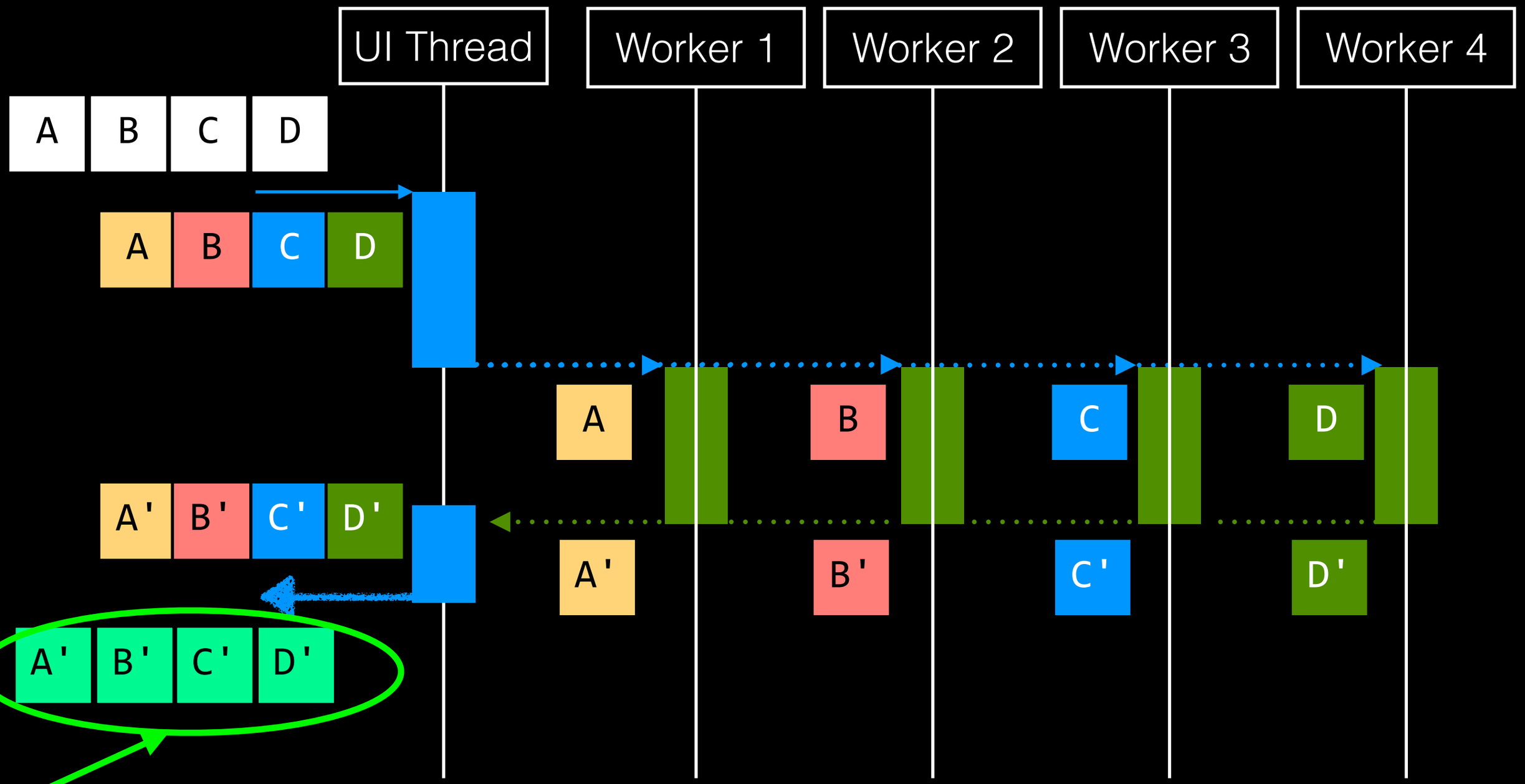
1.000.000 elementos en 4 partes
http://jsperf.com/arraybuffer-split/3

```
function typedArraySlice(array, from, to) {
  return new array.constructor(array.subarray(from, to));
}
```

Partición

Ejemplo

Consolidación de resultados


UI Thread
Worker 1
Worker 2
Worker 3
Worker 4
A
B
C
D
A
B
C
D
A
B
C
D
A'
B'
C'
D'
A'
B'
C'
D'
A'
B'
C'
D'

p-j-s

Navegador	TypedArray.set	Manual
Chrome 40	1.071 ops/sec	327 ops/sec (69% más lento)
FF Nightly 41	1.848 ops/sec	476 ops/sec (74% más lento)

TypedArray

http://jsperf.com/typedarray-merge/2

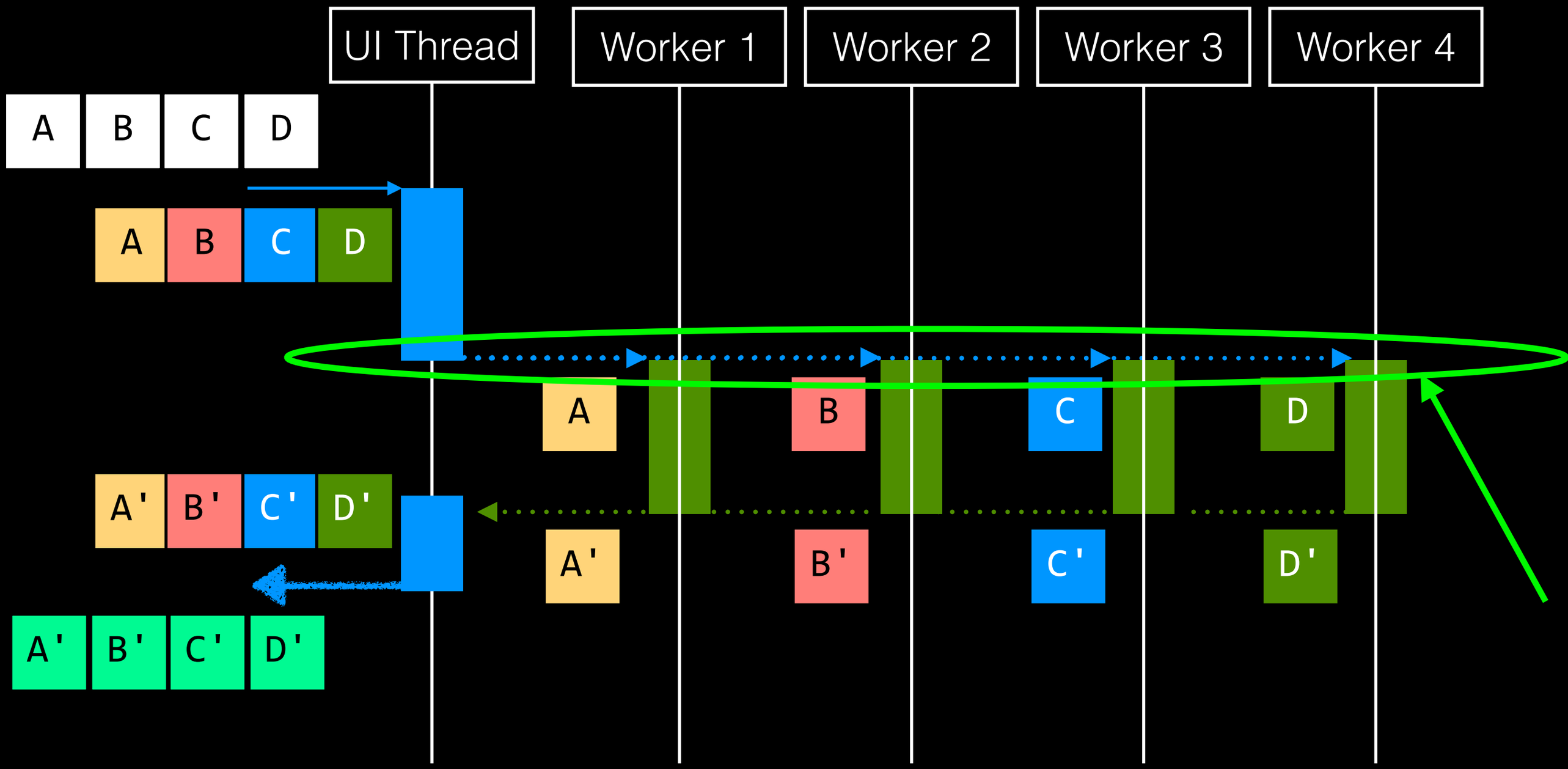
Navegador	TypedArray.set	Manual
FF Nightly 41	340 ops/sec	255 ops/sec (25% más lento)

SharedTypedArray

http://jsperf.com/sharedtypedarray-merge

Optimizaciones

Tiempo transferencia funciones


UI Thread
Worker 1
Worker 2
Worker 3
Worker 4
A
B
C
D
A
B
C
D
A
B
C
D
A'
B'
C'
D'
A'
B'
C'
D'
A'
B'
C'
D'

p-j-s

```
var promise = pjs.updateContext({
  add: x => x + 2,
}).then(() => {
   pjs(new Uint8Array([1,2,3,4]))
     .map('add')
     .seq()
     .then(result => {
       // result is [ 3, 4, 5, 6 ]
     });
});
```

Funciones por clave

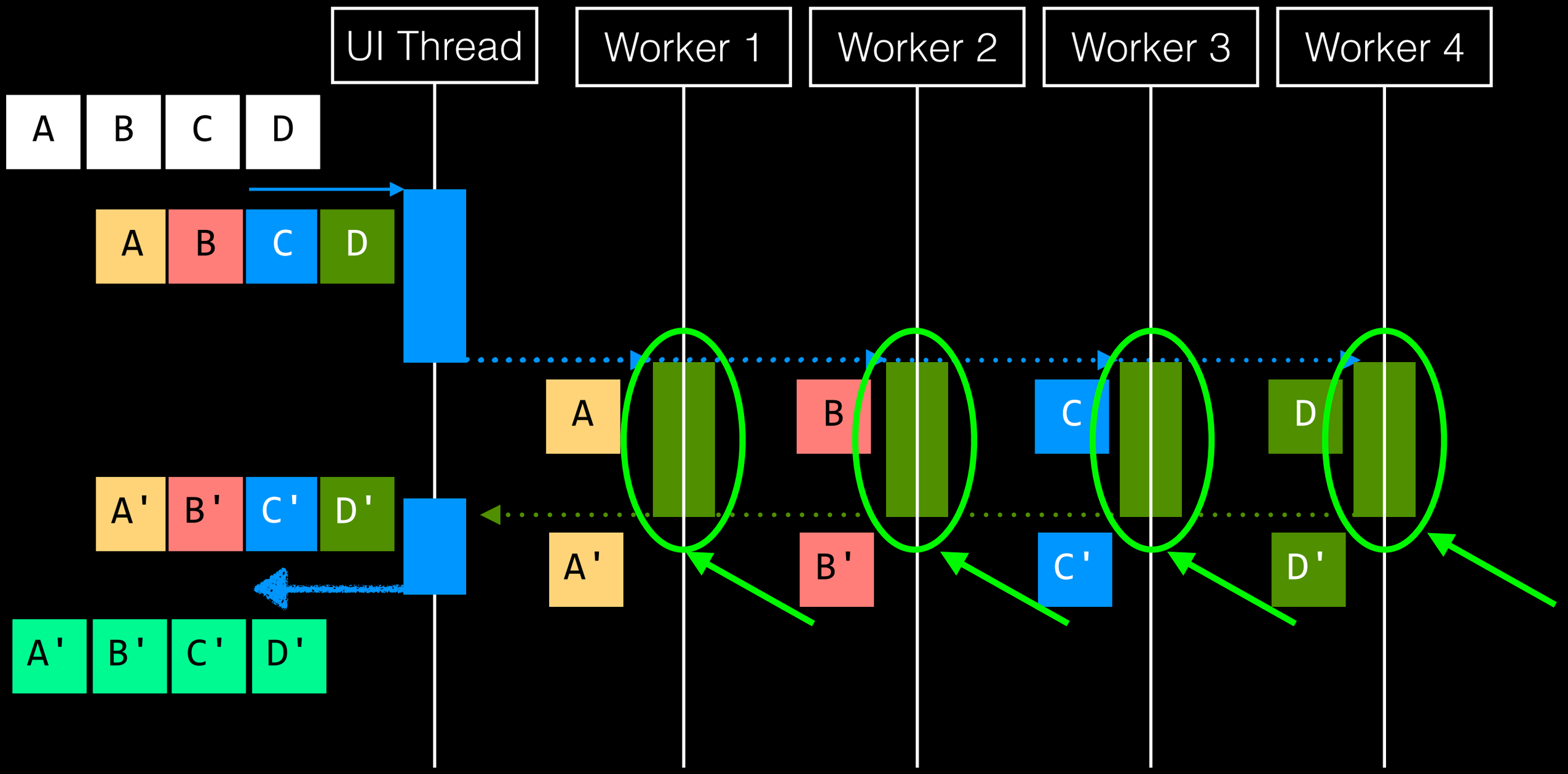
Ejemplo

Navegador	Por nombre	Instancia función
Chrome 40	209 ops/sec	199 ops/sec (5% más lento)
FF Nightly 41	126 ops/sec	126 ops/sec

Funciones por clave

http://jsperf.com/global-ctx-inlined-func/3

Inlining


UI Thread
Worker 1
Worker 2
Worker 3
Worker 4
A
B
C
D
A
B
C
D
A
B
C
D
A'
B'
C'
D'
A'
B'
C'
D'
A'
B'
C'
D'

p-j-s

```
function noise() {
  return Math.random() * 0.5 + 0.5;
};

function clamp(component) {
  return Math.max(Math.min(255, component), 0);
}

function distance(scale, dest, src) {
  return clamp(scale * dest + (1 - scale) * src);
};

function map(data, len) {
  for (var i = 0; i < len; i += 4) {
      var r = data[i];
      var g = data[i + 1];
      var b = data[i + 2];

      data[i] = distance(noise(), (r * 0.393) + (g * 0.769) + (b * 0.189), r);
      data[i + 1] = distance(noise(), (r * 0.349) + (g * 0.686) + (b * 0.168), g);
      data[i + 2] = distance(noise(), (r * 0.272) + (g * 0.534) + (b * 0.131), b);
  }
};
```

Tono sepia

Secuencial

```
pjs(new Uint32Array(canvasData.data.buffer)).map(pixel => {
  function noise() {
      return Math.random() * 0.5 + 0.5;
  };

  function clamp(component) {
      return Math.max(Math.min(255, component), 0);
  }

  function distance(scale, dest, src) {
      return clamp(scale * dest + (1 - scale) * src);
  };

  var r = pixel & 0xFF;
  var g = (pixel & 0xFF00) >> 8;
  var b = (pixel & 0xFF0000) >> 16;
  var new_r = distance(noise(), (r * 0.393) + (g * 0.769) + (b * 0.189), r);
  var new_g = distance(noise(), (r * 0.349) + (g * 0.686) + (b * 0.168), g);
  var new_b = distance(noise(), (r * 0.272) + (g * 0.534) + (b * 0.131), b);

  return (pixel & 0xFF000000) + (new_b << 16) + (new_g << 8) + (new_r & 0xFF);
}).seq(result => { /* trabajar con el resultado */ });
```

Tono sepia

Paralelo

```
pjs(new Uint32Array(canvasData.data.buffer)).map(pixel => {
  function noise() {
      return Math.random() * 0.5 + 0.5;
  };

  function clamp(component) {
      return Math.max(Math.min(255, component), 0);
  }

  function distance(scale, dest, src) {
      return clamp(scale * dest + (1 - scale) * src);
  };

  var r = pixel & 0xFF;
  var g = (pixel & 0xFF00) >> 8;
  var b = (pixel & 0xFF0000) >> 16;
  var new_r = distance(noise(), (r * 0.393) + (g * 0.769) + (b * 0.189), r);
  var new_g = distance(noise(), (r * 0.349) + (g * 0.686) + (b * 0.168), g);
  var new_b = distance(noise(), (r * 0.272) + (g * 0.534) + (b * 0.131), b);

  return (pixel & 0xFF000000) + (new_b << 16) + (new_g << 8) + (new_r & 0xFF);
}).seq(result => { /* trabajar con el resultado */ });
```

Tono sepia

Paralelo

```
pjs(buff).map(pixel => {
  var r = pixel & 0xFF;
  var g = (pixel & 0xFF00) >> 8;
  var b = (pixel & 0xFF0000) >> 16;
  var noise_r = Math.random() * 0.5 + 0.5;
  var noise_g = Math.random() * 0.5 + 0.5;
  var noise_b = Math.random() * 0.5 + 0.5;

  var new_r = Math.max(Math.min(255, noise_r * ((r * 0.393) + (g * 0.769) + (b * 0.189))
+ (1 - noise_r) * r), 0);
  var new_g = Math.max(Math.min(255, noise_g * ((r * 0.349) + (g * 0.686) + (b * 0.168))
+ (1 - noise_g) * g), 0);
  var new_b = Math.max(Math.min(255, noise_b * ((r * 0.272) + (g * 0.534) + (b * 0.131))
+ (1 - noise_b) * b), 0);

  return (pixel & 0xFF000000) + (new_b << 16) + (new_g << 8) + (new_r & 0xFF);
}.seq(result => { /* trabajar con resultado */ });
```

Tono sepia

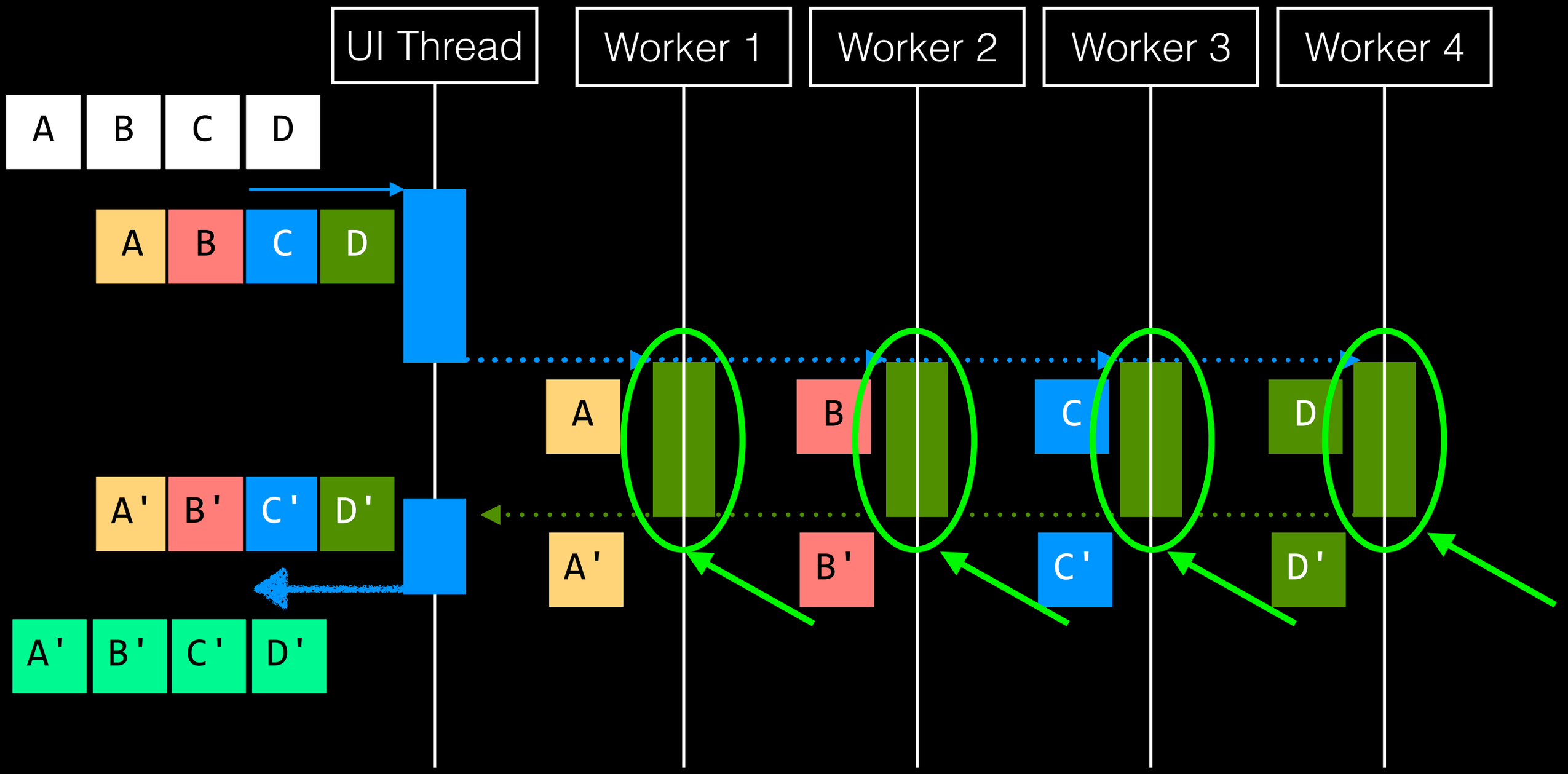
Paralelo

Navegador	Manual inlining	Sin inlining
Chrome 40	27 ops/sec	6 ops/sec (78% más lento)
FF Nightly 41	24 ops/sec	15 ops/sec (38% más lento)

Inlining

http://jsperf.com/pjs-map-inlining/7

Caché de funciones


UI Thread
Worker 1
Worker 2
Worker 3
Worker 4
A
B
C
D
A
B
C
D
A
B
C
D
A'
B'
C'
D'
A'
B'
C'
D'
A'
B'
C'
D'

p-j-s

Navegador	Con caché	Sin caché
Chrome 40	26 ops/sec	26 ops/sec
FF Nightly 41	25 ops/sec	21 ops/sec (16% más lento)

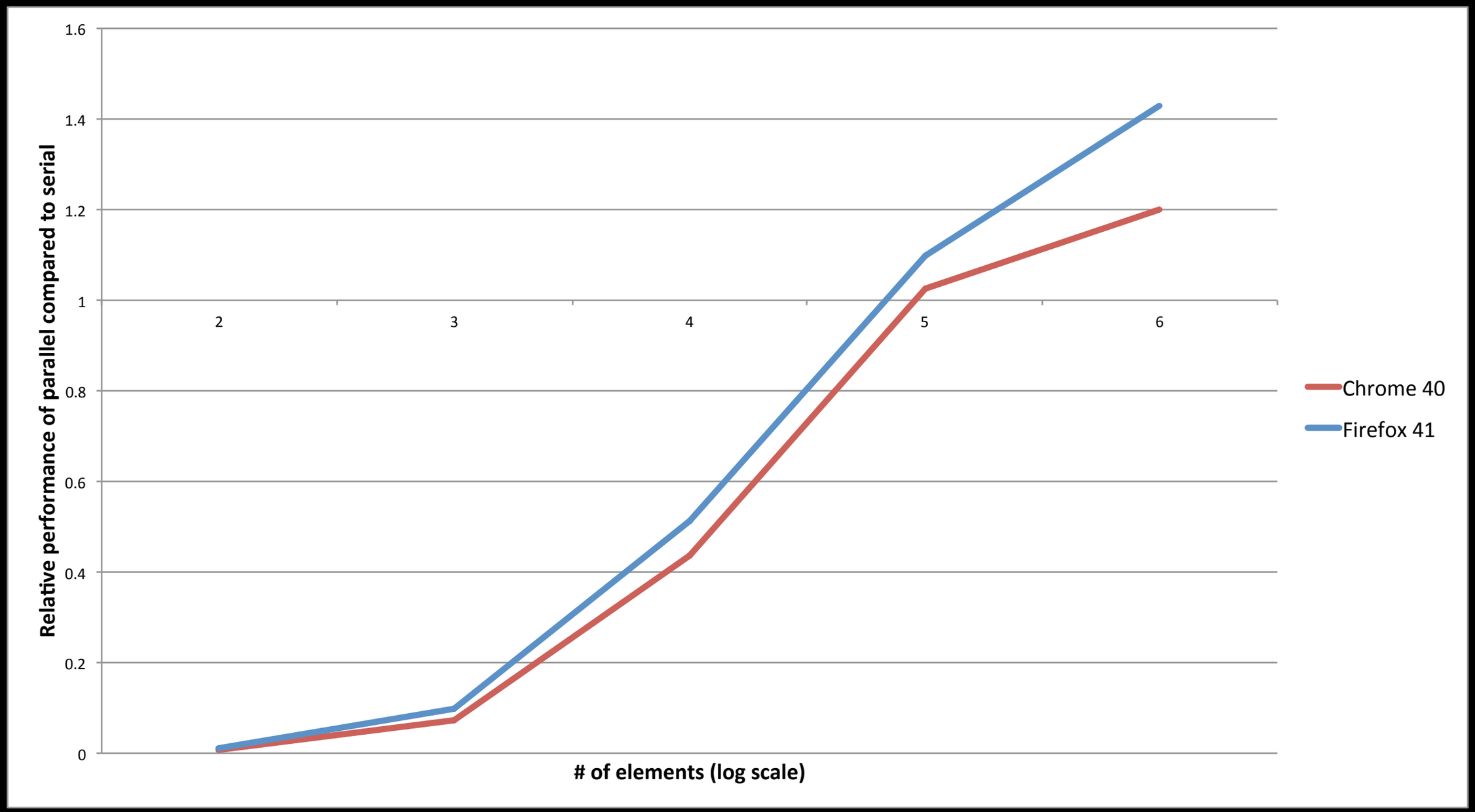
Caché de funciones

http://jsperf.com/p-j-s-with-vs-without-function-cache/4

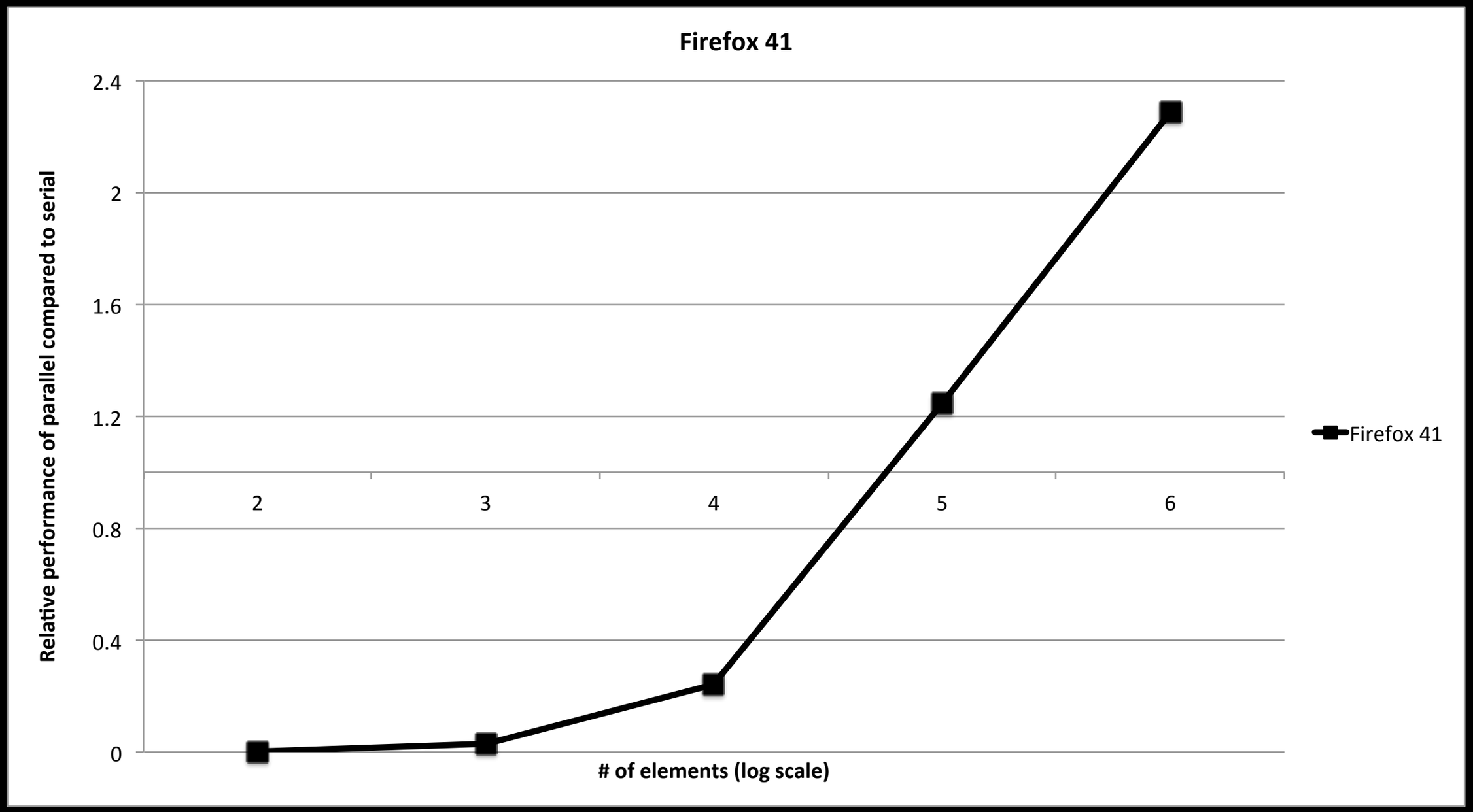
Case de estudio

Transformación de imágenes

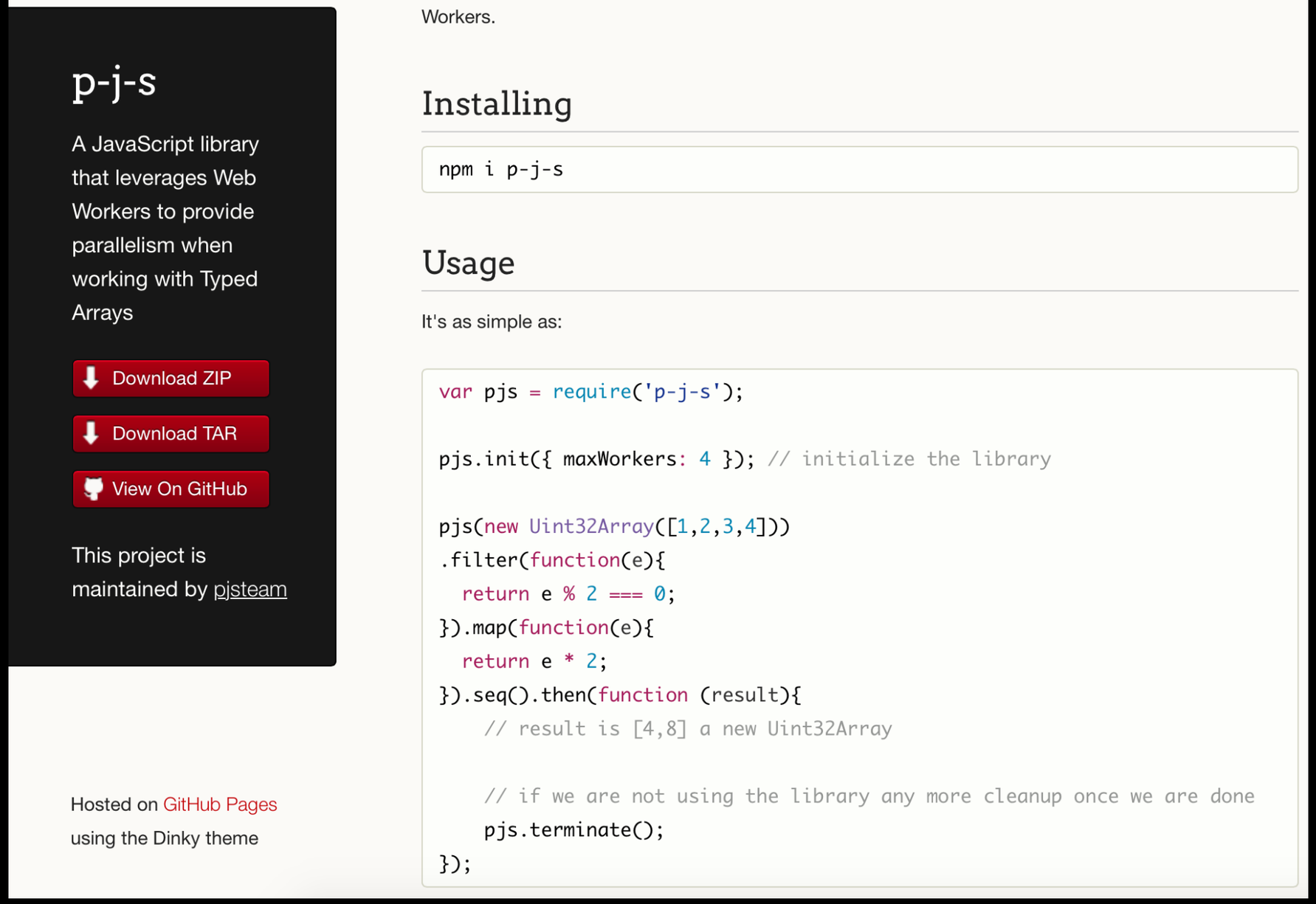
Demo


1.6
1.4
1.2
1
0.8
0.6
0.4
0.2
0
Relative performance of parallel compared to serial
2
3
4
5
6
# of elements (log scale)
Chrome 40
Firefox 41

TypedArray



SharedTypedArray



Sitio web

http://pjsteam.github.io/pjs/

Entregables

- Paper propuesto para CARLA 2015
- Reporte técnico con benchmarks y decisiones de diseño
- Repositorio open source con el código de la biblioteca y documentación de uso
- Sitio web para la biblioteca: http://pjsteam.github.io/pjs/
- Charla en JSConfUY en la que se presentó la biblioteca

Preguntas?

Gracias!