Faster Applications

Minko Gechev

rhyme

Minko Gechev

mgechev

Functional time traveler.

California, USA

http://blog.mgechev.com/

Organizations

Overview | Repositories 210 | Stars 99 | Followers 3.6k

angular-style-guide

Set of best practices for Angular application development

5k 744

angular-seed

Modular starter project for Angular 2 (and beyond) with statically typed build and AoT compilation

TypeScript 4.4k 1.7k

codelyzer

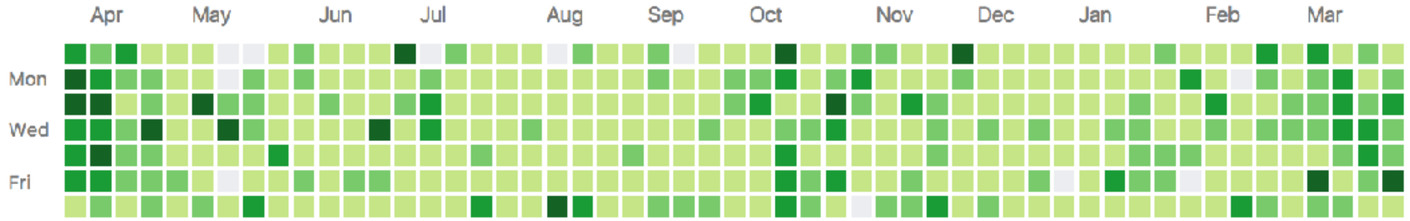
Linting for Angular projects.

TypeScript 1.4k 133

angular/mobile-toolkit

JavaScript implementation of different computer science algorithms.

TypeScript 1,104 150



twitter.com/**mgechev**

github.com/**mgechev**


Minko Gechev
Foreword by Miško Hevery, Creator of AngularJS and Angular
Switching to Angular
Third Edition
Align with Google's long-term vision for Angular versi
and beyond
Minko Gechev

Runtime Performance

Simplified

Business application

twitter.com/mgechev

Purely Fast

Minko

localhost:5557

Sales

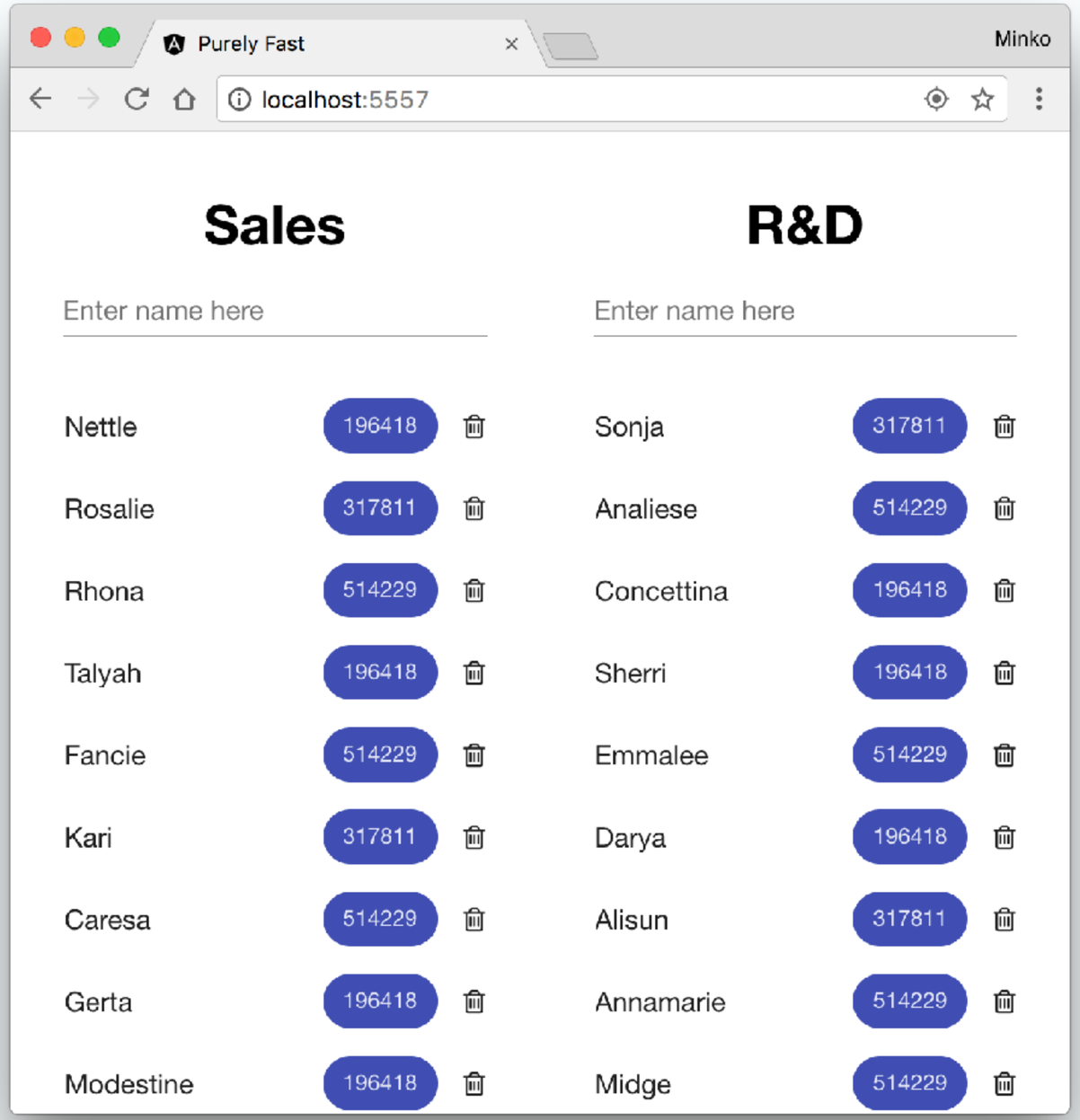
Enter name here

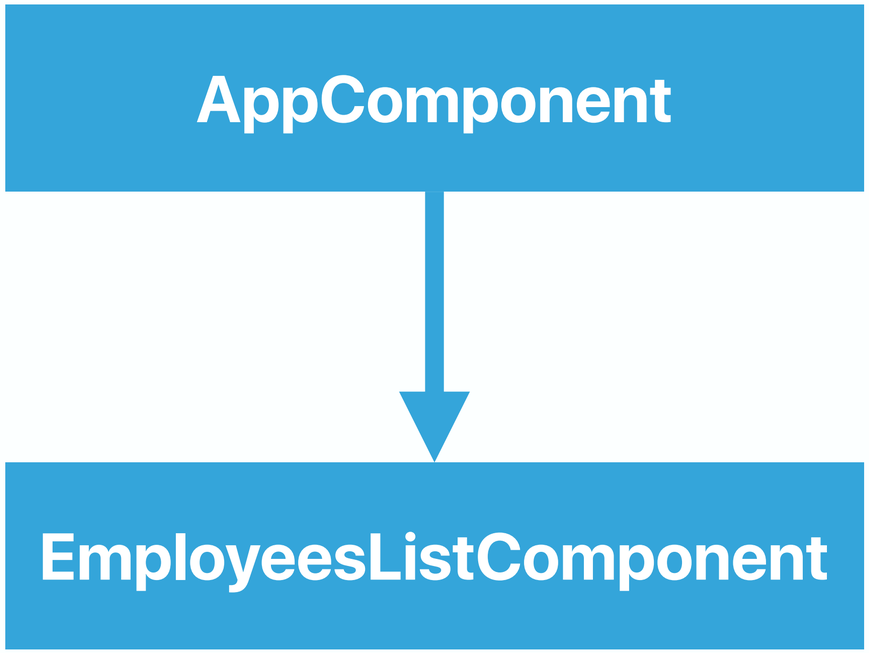
Cate	75025
Jeanna	75025
Lisbeth	196418
Neda	75025
Angel	28657
Correna	28657
Ingeborg	75025

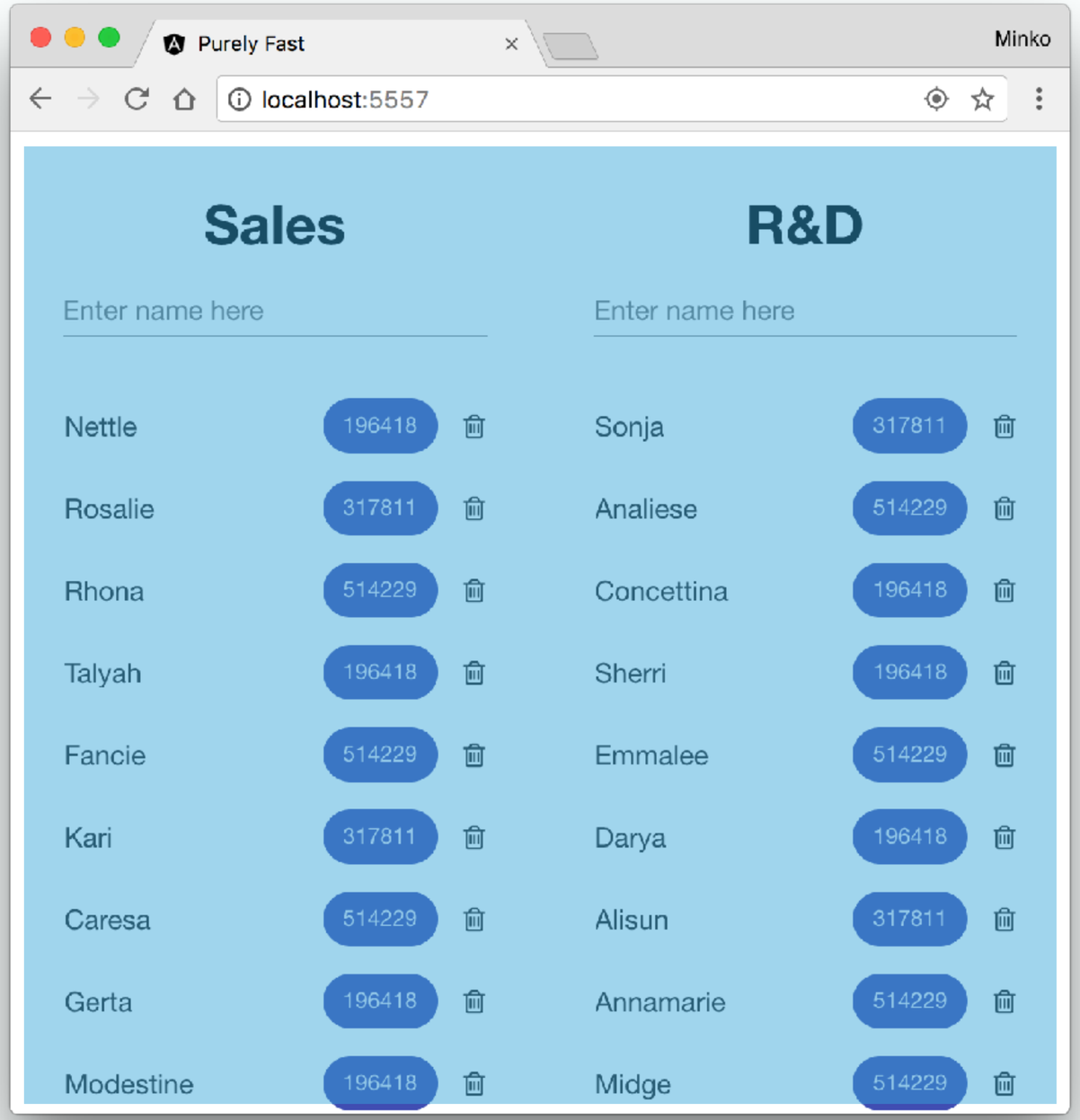
R&D

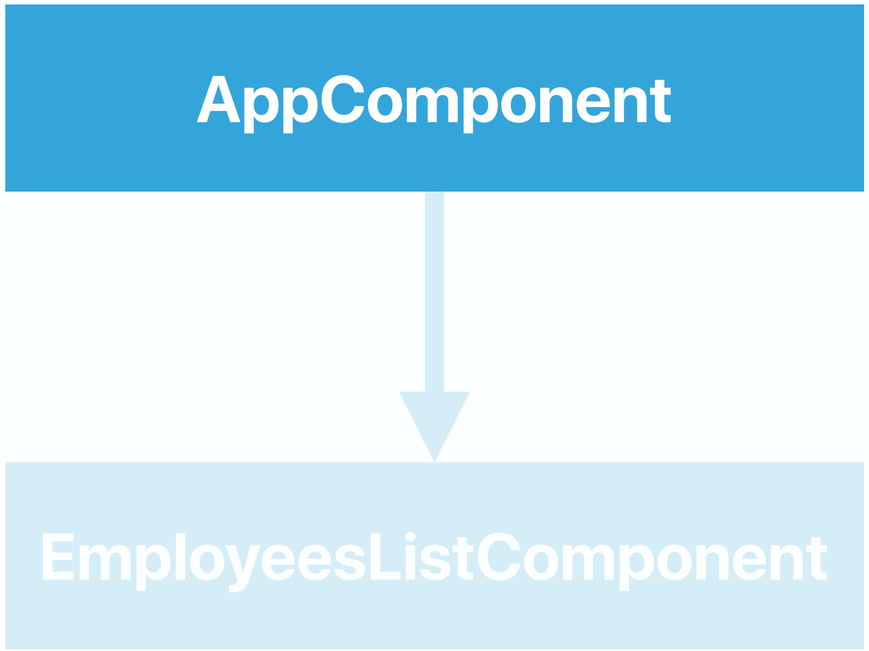
Enter name here

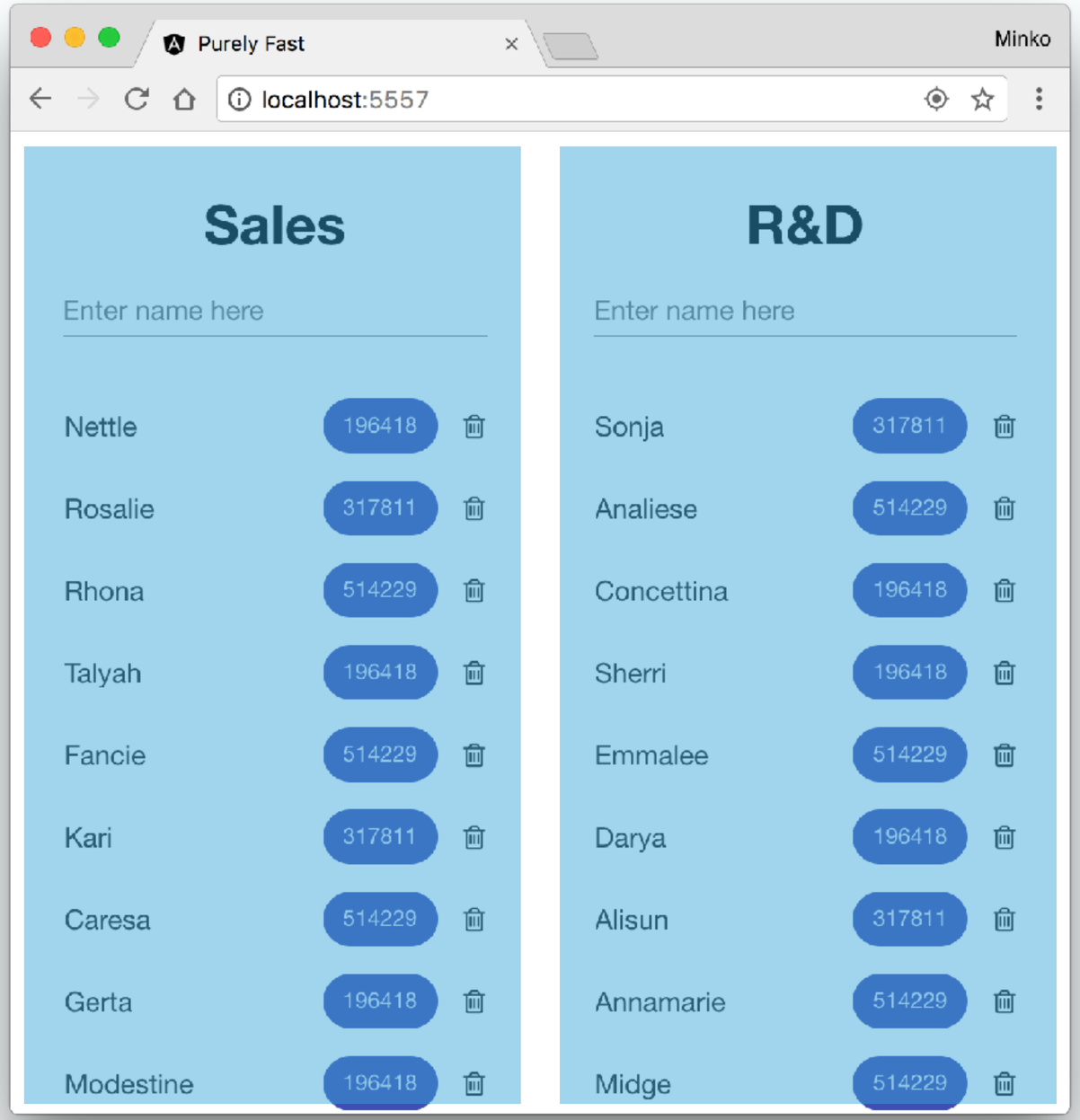
Kimberley	196418
Marlie	28657
Marjorie	75025
Benetta	121393
Ruthann	28657
Sharyl	196418
Coriss	196418

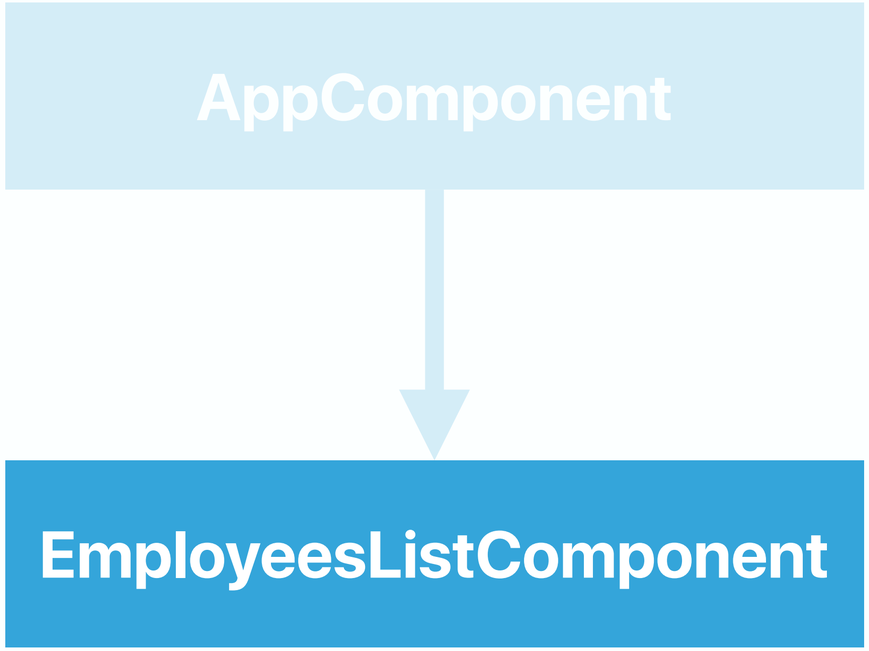

Purely Fast
Minko
localhost:5557
Sales
R&D
Enter name here
Enter name here
Nettle 196418
Rosalie 317811
Rhona 514229
Talyah 196418
Fancie 514229
Kari 317811
Caresa 514229
Gerta 196418
Modestine 196418
Sonja 317811
Analiese 514229
Concettina 196418
Sherri 196418
Emmalee 514229
Darya 196418
Alisun 317811
Annamarie 514229
Midge 514229


AppComponent
EmployeesListComponent


Purely Fast
Minko
localhost:5557
Sales
R&D
Enter name here
Enter name here
Nettle 196418
Rosalie 317811
Rhona 514229
Talyah 196418
Fancie 514229
Kari 317811
Caresa 514229
Gerta 196418
Modestine 196418
Sonja 317811
Analiese 514229
Concettina 196418
Sherri 196418
Emmalee 514229
Darya 196418
Alisun 317811
Annamarie 514229
Midge 514229


AppComponent
EmployeesListComponent


Purely Fast
Minko
localhost:5557
Sales
Enter name here
Nettle 196418
Rosalie 317811
Rhona 514229
Talyah 196418
Fancie 514229
Kari 317811
Caresa 514229
Gerta 196418
Modestine 196418
R&D
Enter name here
Sonja 317811
Analiese 514229
Concettina 196418
Sherri 196418
Emmalee 514229
Darya 196418
Alisun 317811
Annamarie 514229
Midge 514229


AppComponent
EmployeesListComponent

Sales

Enter name here

Jeanna	75025
Carla	196418
Ezmeralda	121393
Karena	196418
Birdie	28657
Angy	75025
Rennie	121393
Lidia	46368

```
<input [(ngModel)]="label"
  (keydown)="handleKey($event)">

<mat-list-item
  *ngFor="let item of data”>

  {{ item.label }}

  {{ calculate(item.num) }}

</mat-list-item>
```

Sales

Enter name here

Jeanna	75025
Carla	196418
Ezmeralda	121393
Karena	196418
Birdie	28657
Angy	75025
Rennie	121393
Lidia	46368

```
<input [(ngModel)]="label"
  (keydown)="handleKey($event)">

<mat-list-item
  *ngFor="let item of data”>

  {{ item.label }}

  {{ calculate(item.num) }}

</mat-list-item>
```

Sales

Enter name here

Jeanna	75025
Carla	196418
Ezmeralda	121393
Karena	196418
Birdie	28657
Angy	75025
Rennie	121393
Lidia	46368

```
<input [(ngModel)]="label"
  (keydown)="handleKey($event)">

<mat-list-item
  *ngFor="let item of data”>

  {{ item.label }}

  {{ calculate(item.num) }}

</mat-list-item>
```

Sales

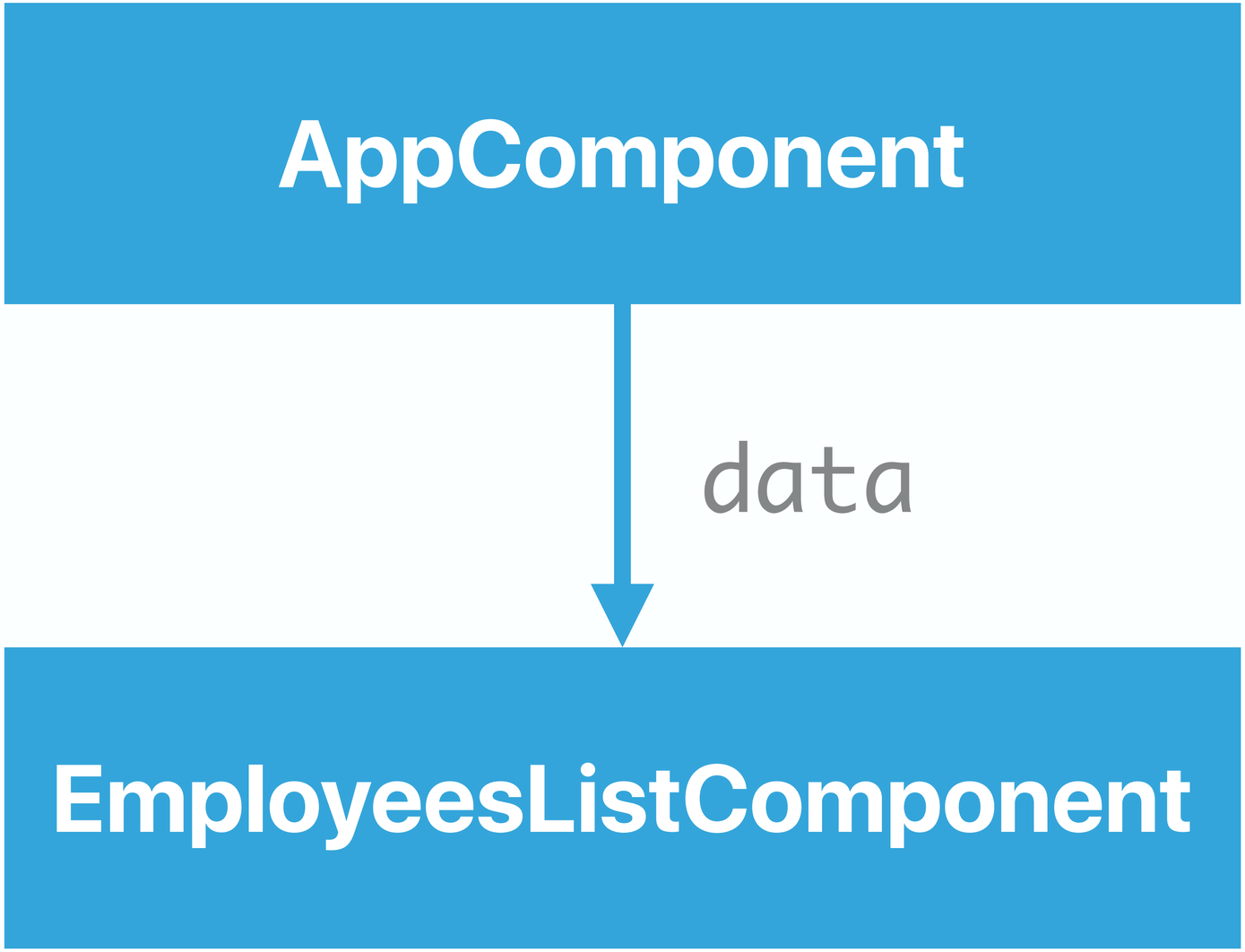
Enter name here

Jeanna	75025
Carla	196418
Ezmeralda	121393
Karena	196418
Birdie	28657
Angy	75025
Rennie	121393
Lidia	46368

```
<input [(ngModel)]="label"
  (keydown)="handleKey($event)">

<mat-list-item
  *ngFor="let item of data”>

  {{ item.label }}

  {{ calculate(item.num) }}

</mat-list-item>
```

```
@Component(...)
export class EmployeeListComponent {
  @Input() data: EmployeeData[];

  @Output() remove = new EventEmitter<EmployeeData>();
  @Output() add = new EventEmitter<string>();

  handleKey(event: any) { ... }

  calculate(num: number) {
    return fibonacci(num);
  }
}
```

```
@Component(...)
export class EmployeeListComponent {
  @Input() data: EmployeeData[];

  @Output() remove = new EventEmitter<EmployeeData>();
  @Output() add = new EventEmitter<string>();

  handleKey(event: any) { ... }

  calculate(num: number) {
    return fibonacci(num);
  }
}
```


AppComponent
data
EmployeesListComponent

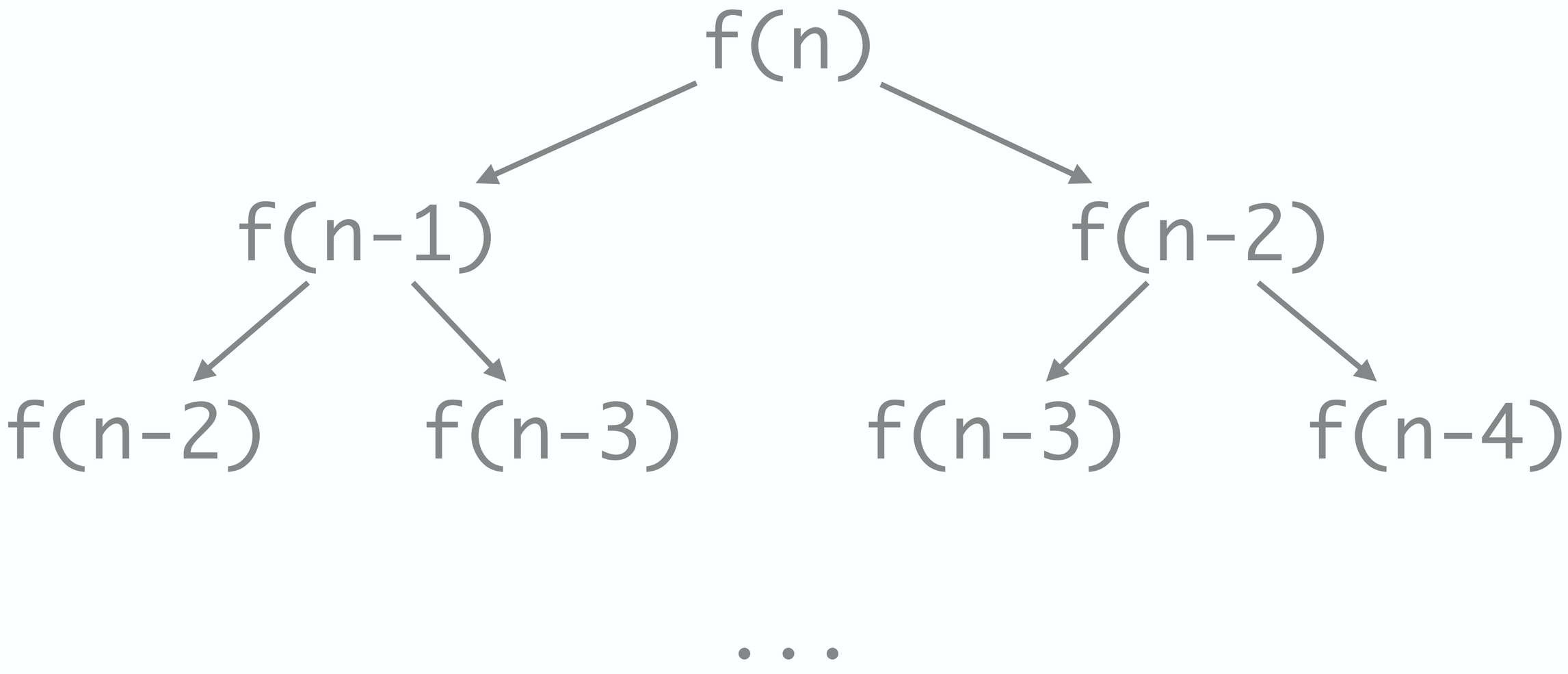
```
@Component(...)
export class EmployeeListComponent {
  @Input() data: EmployeeData[];

  @Output() remove = new EventEmitter<EmployeeData>();
  @Output() add = new EventEmitter<string>();

  handleKey(event: any) { ... }

  calculate(num: number) {
    return fibonacci(num);
  }
}
```

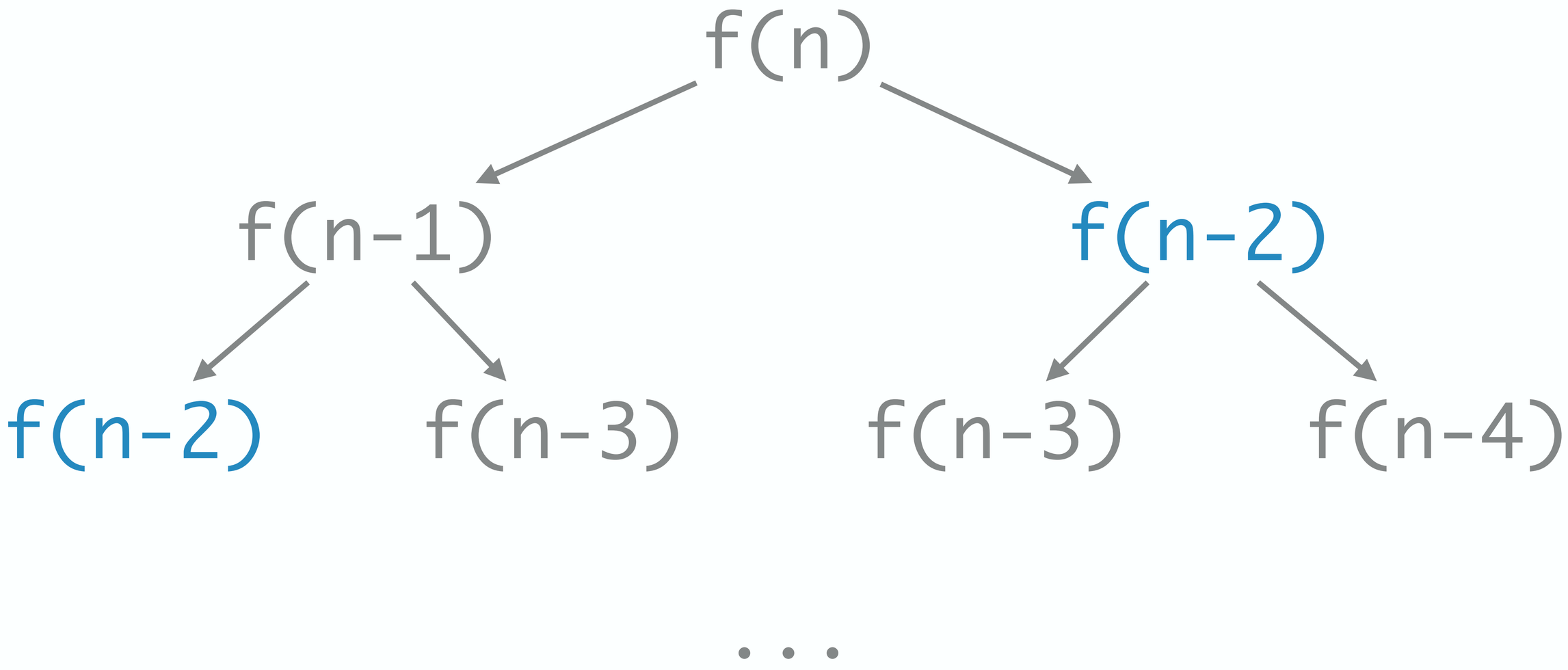
Sales

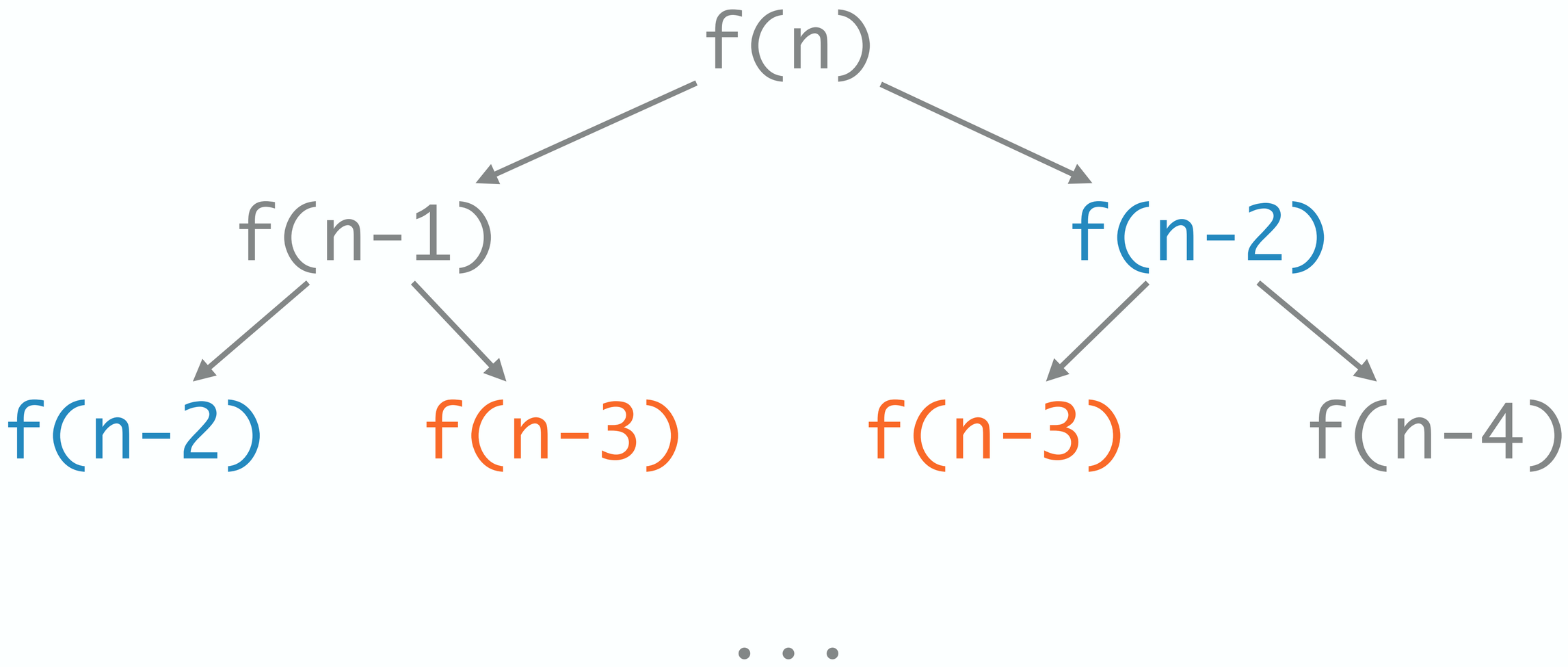
Enter name here

Jeanna	75025
Carla	196418
Ezmeralda	121393
Karena	196418
Birdie	28657
Angy	75025
Rennie	121393
Lidia	46368

```
const fibonacci = n => {
  if (n === 1 || n === 2)
    return 1;
  return fibonacci(n - 1)
       + fibonacci(n - 2);
};
```


f(n)
f(n-1)
f(n-2)
f(n-2)
f(n-3)
f(n-3)
f(n-4)
...


f(n)
f(n-1)
f(n-2)
f(n-2)
f(n-3)
f(n-3)
f(n-4)
...


f(n)
f(n-1)
f(n-2)
f(n-2)
f(n-3)
f(n-3)
f(n-4)
. . .

Slowing it down artificially

Application Structure

- An application component
- Two list components
  - Slow computation for each entry

Real data...

Purely Fast

Minko

localhost:5557

Sales

Enter name here

Nettle	196418
Rosalie	317811
Rhona	514229
Talyah	196418
Fancie	514229
Kari	317811
Caresa	514229
Gerta	196418
Modestine	196418

R&D

Enter name here

Sonja	317811
Analiese	514229
Concettina	196418
Sherri	196418
Emmalee	514229
Darya	196418
Alisun	317811
Annamarie	514229
Midge	514229

Purely Fast

Minko

localhost:5557

Sales

Enter name here

Nettle	196418
Rosalie	317811
Rhona	514229
Talyah	196418
Fancie	514229
Kari	317811
Caresa	514229
Gerta	196418
Modestine	196418

R&D

Enter name here

Sonja	317811
Analiese	514229
Concettina	196418
Sherri	196418
Emmalee	514229
Darya	196418
Alisun	317811
Annamarie	514229
Midge	514229

140 entries

Purely Fast

Minko

localhost:5557

Sales

Enter name here

Nettle	196418
Rosalie	317811
Rhona	514229
Talyah	196418
Fancie	514229
Kari	317811
Caresa	514229
Gerta	196418
Modestine	196418

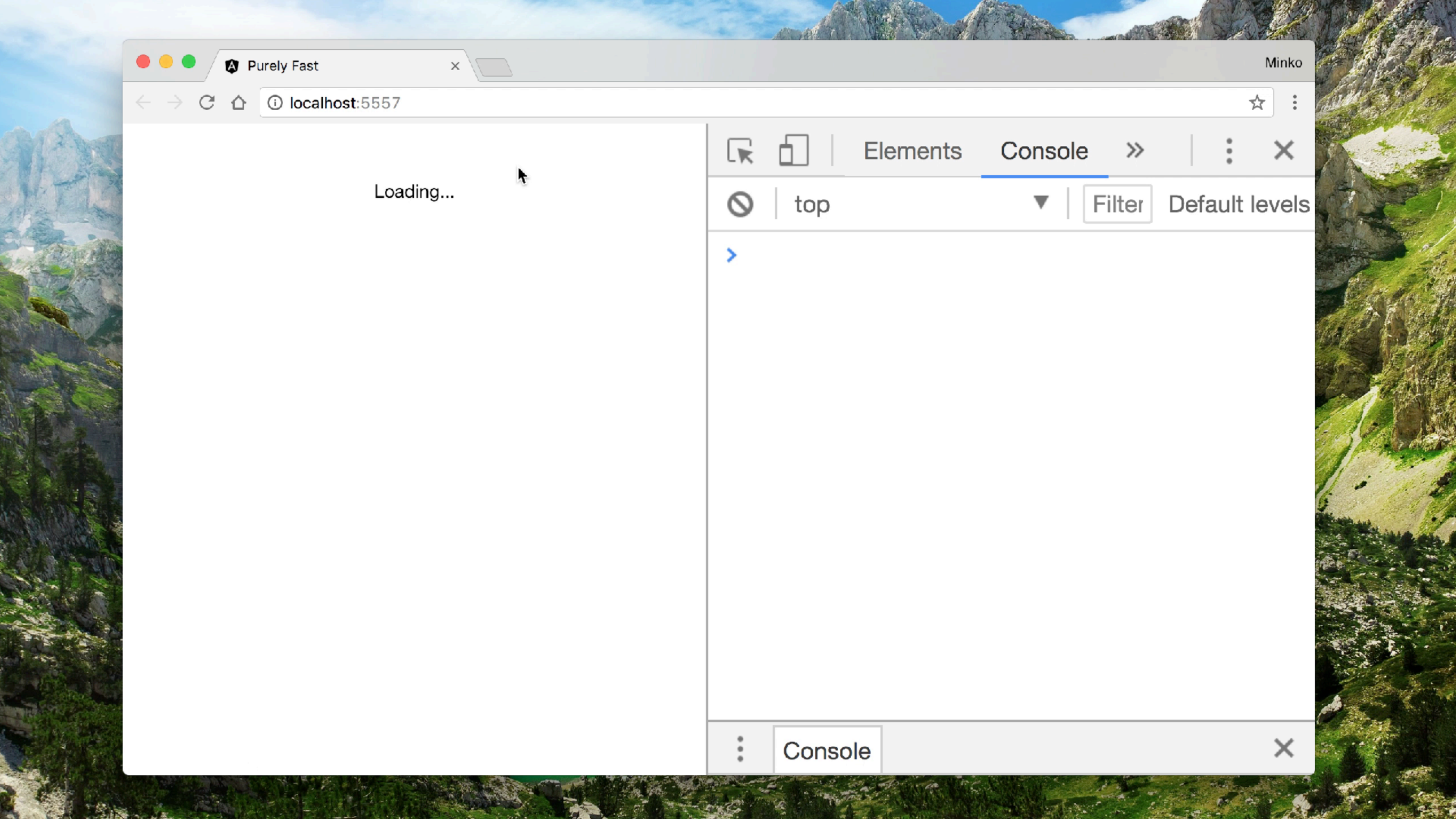
R&D

Enter name here

Sonja	317811
Analiese	514229
Concettina	196418
Sherri	196418
Emmalee	514229
Darya	196418
Alisun	317811
Annamarie	514229
Midge	514229

Why that slow?

```
@Component(...)
export class EmployeeListComponent {
  ...
  calculate(num: number) {
    console.log('Computing for entry in',
        this.department);
    return fibonacci(num);
  }
}
```


Purely Fast
Minko
localhost:5557
Loading...
Elements
Console
top
Filter
Default levels
Console

Ideas for optimization?

OnPush

Change Detection Strategy

twitter.com/mgechev

With `OnPush` change detection will be triggered when the framework, with reference check, determines that any of the inputs of a component has changed...

What does this mean?

Lets think of EmployeeListComponent as a function, where:

- Inputs are function's arguments
- Rendered component is function's result

Pseudo code (not Angular)

```
const f = EmployeeListComponent;
const data = [e1];

// Will trigger CD
f({ data: data });

data.push(e2);
// Will not trigger CD
f({ data: data });

// Will trigger CD
f({ data: data.slice() });
```

Pseudo code (not Angular)

```
const f = EmployeeListComponent;
const data = [e1];

// Will trigger CD
f({ data: data });

data.push(e2);
// Will not trigger CD
f({ data: data });

// Will trigger CD
f({ data: data.slice() });
```

Pseudo code (not Angular)

```
const f = EmployeeListComponent;
const data = [e1];

// Will trigger CD
f({ data: data });

data.push(e2);
// Will not trigger CD
f({ data: data });

// Will trigger CD
f({ data: data.slice() });
```

Pseudo code (not Angular)

```
const f = EmployeeListComponent;
const data = [e1];

// Will trigger CD
f({ data: data });

data.push(e2);
// Will not trigger CD
f({ data: data });

// Will trigger CD
f({ data: data.slice() });
```

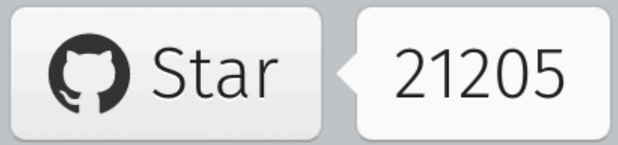
Passing new reference

triggers the change detection

Should we copy the array every time?

Why would we do that...?

IMMUTABLE



Immutable.js helps:

- We get a new reference on change
- We do not copy the entire data structure

```
@Component({
  template: `
    <sd-employee-list
      [data]="list"
      (add)="list = add(list, $event)"
      (remove)="list = remove(list, $event)"
    ></sd-employee-list>
  `
})
export class AppComponent implements OnInit {
  list: List<EmployeeData>;

  add(list: List<EmployeeData>, name: string) {
    return list.unshift({ label: name, num: ... });
  }

  remove(list: List<EmployeeData>, node: EmployeeData) {
    return list.splice(list.indexOf(node), 1);
  }
}
```

```
@Component({
  template: `
    <sd-employee-list
      [data]="list"
      (add)="list = add(list, $event)"
      (remove)="list = remove(list, $event)"
    ></sd-employee-list>
  `
})
export class AppComponent implements OnInit {
  list: List<EmployeeData>;

  add(list: List<EmployeeData>, name: string) {
    return list.unshift({ label: name, num: ... });
  }

  remove(list: List<EmployeeData>, node: EmployeeData) {
    return list.splice(list.indexOf(node), 1);
  }
}
```

```
@Component({
  template: `
    <sd-employee-list
      [data]="list"
      (add)="list = add(list, $event)"
      (remove)="list = remove(list, $event)"
    ></sd-employee-list>
  `
})
export class AppComponent implements OnInit {
  list: List<EmployeeData>;

  add(list: List<EmployeeData>, name: string) {
    return list.unshift({ label: name, num: ... });
  }

  remove(list: List<EmployeeData>, node: EmployeeData) {
    return list.splice(list.indexOf(node), 1);
  }
}
```

Lets see how fast it is now!

Purely Fast

Minko

localhost:5557

Sales

Enter name here

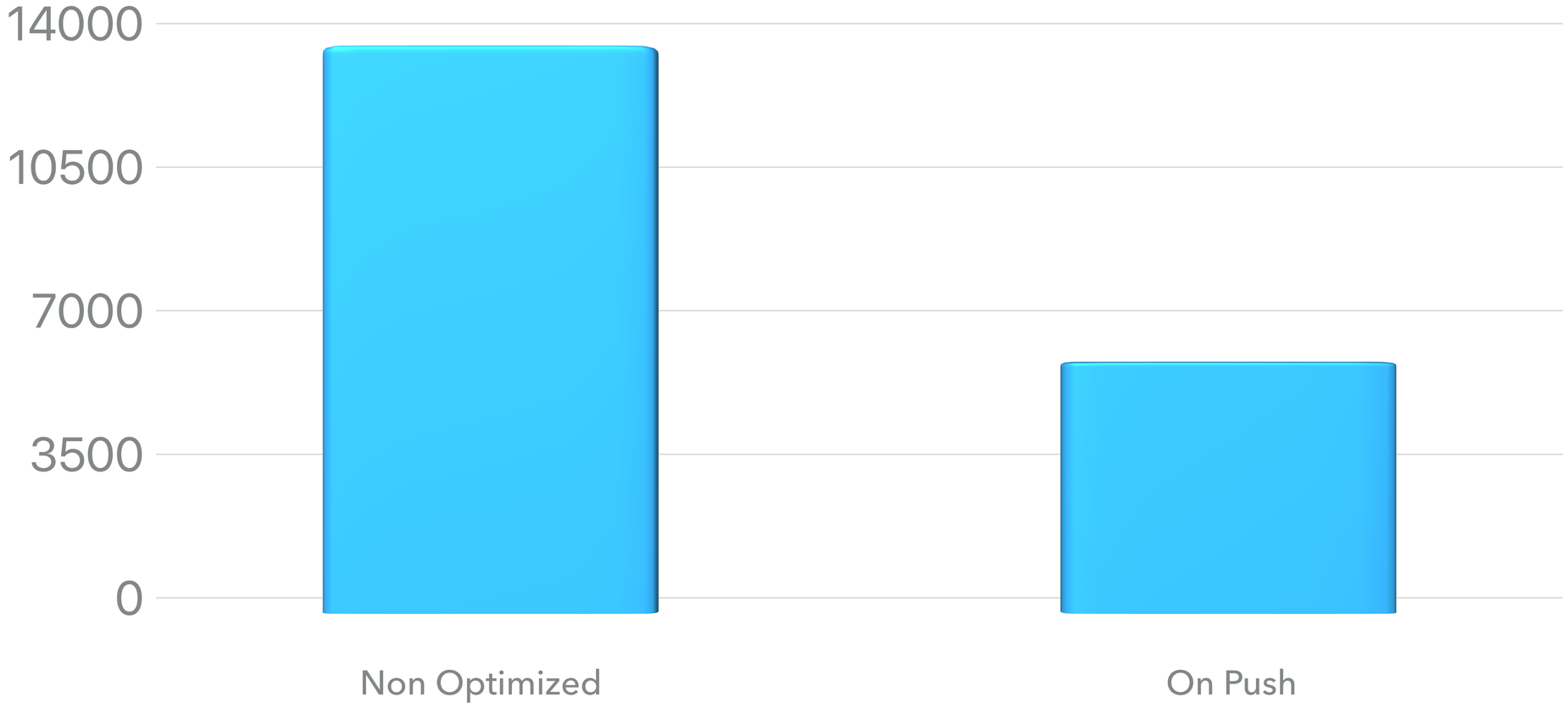
Nettle	196418
Rosalie	317811
Rhona	514229
Talyah	196418
Fancie	514229
Kari	317811
Caresa	514229
Gerta	196418
Modestine	196418

R&D

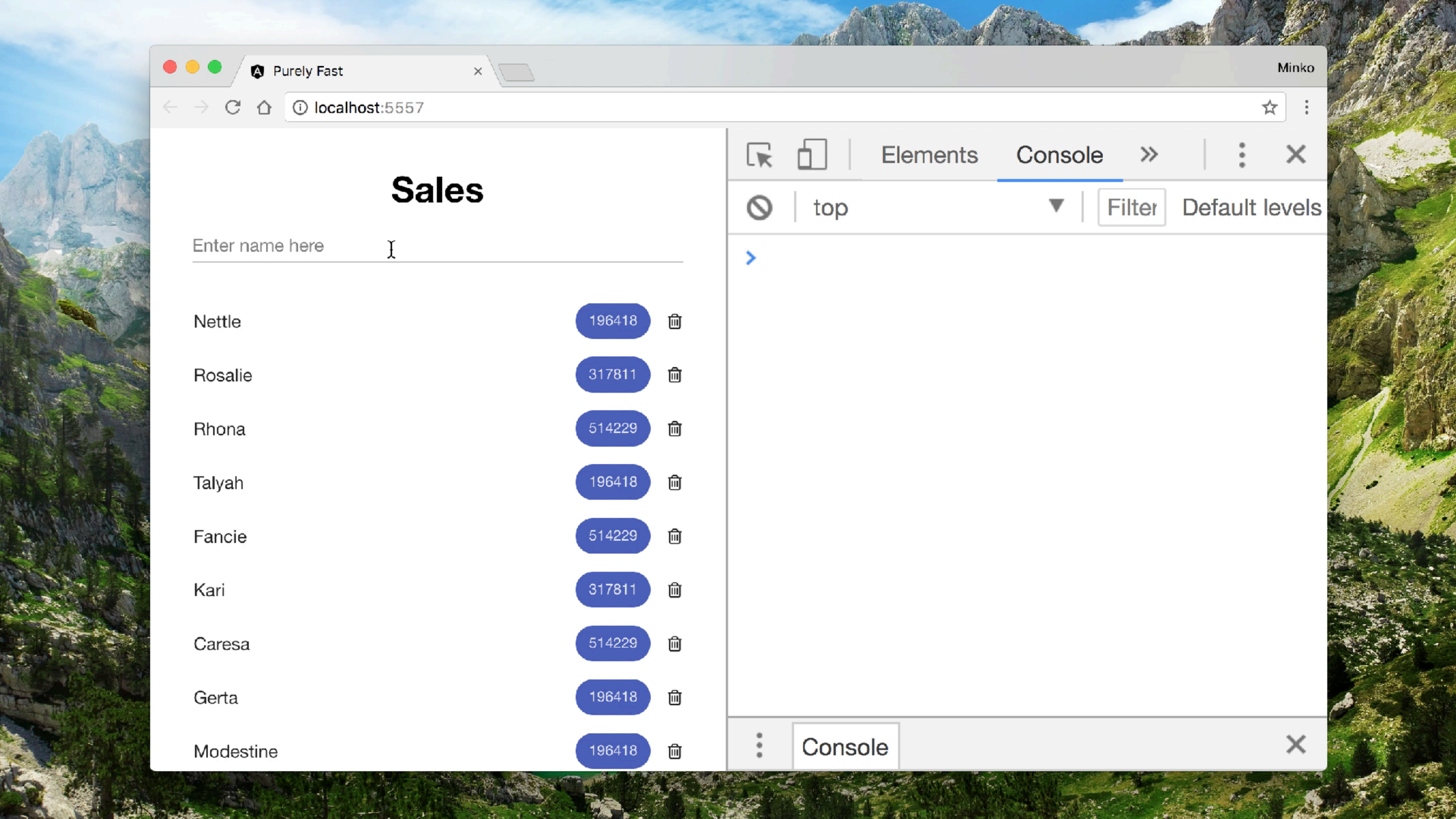
Enter name here

Sonja	317811
Analiese	514229
Concettina	196418
Sherri	196418
Emmalee	514229
Darya	196418
Alisun	317811
Annamarie	514229
Midge	514229

Typing Speed

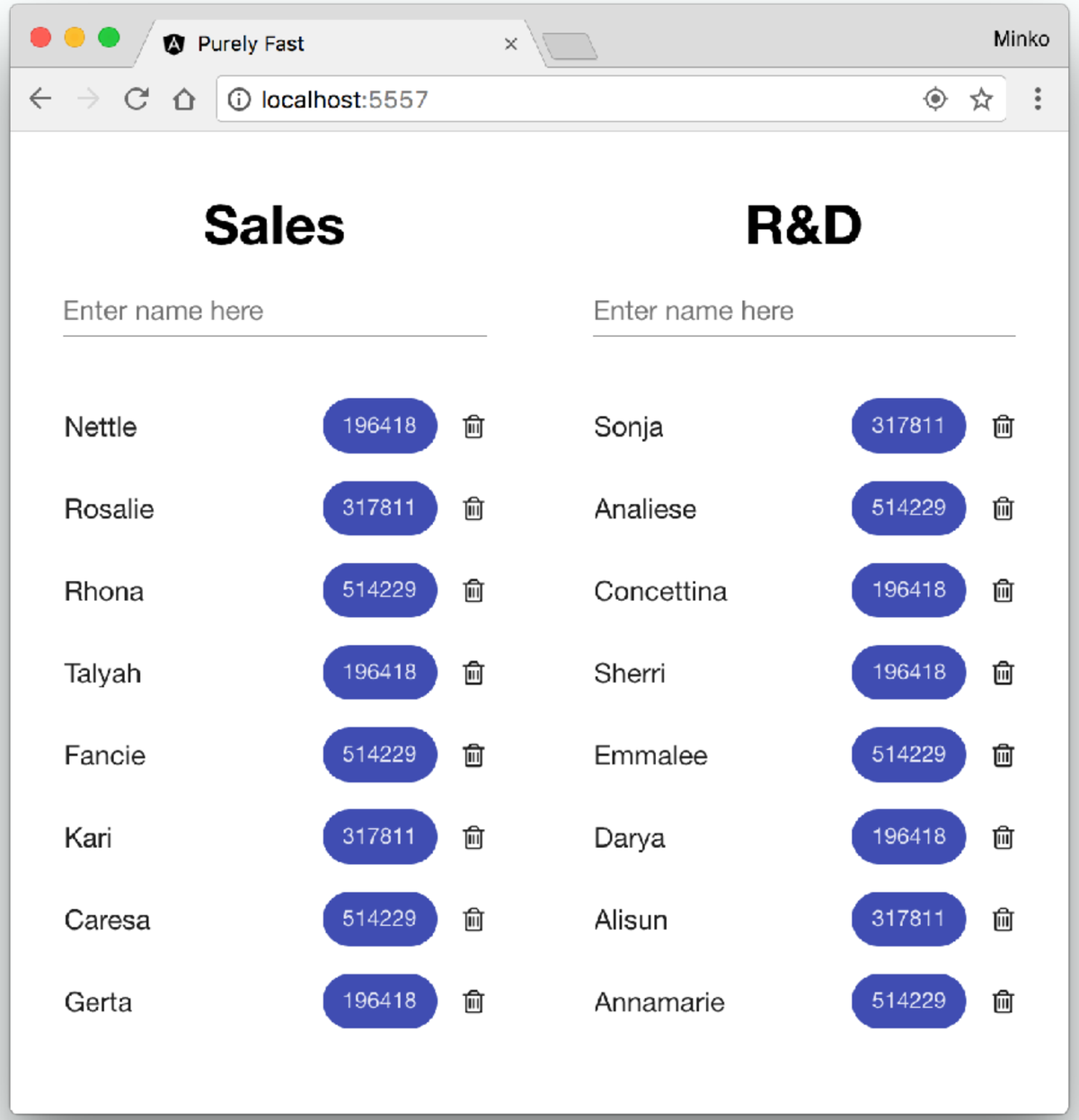


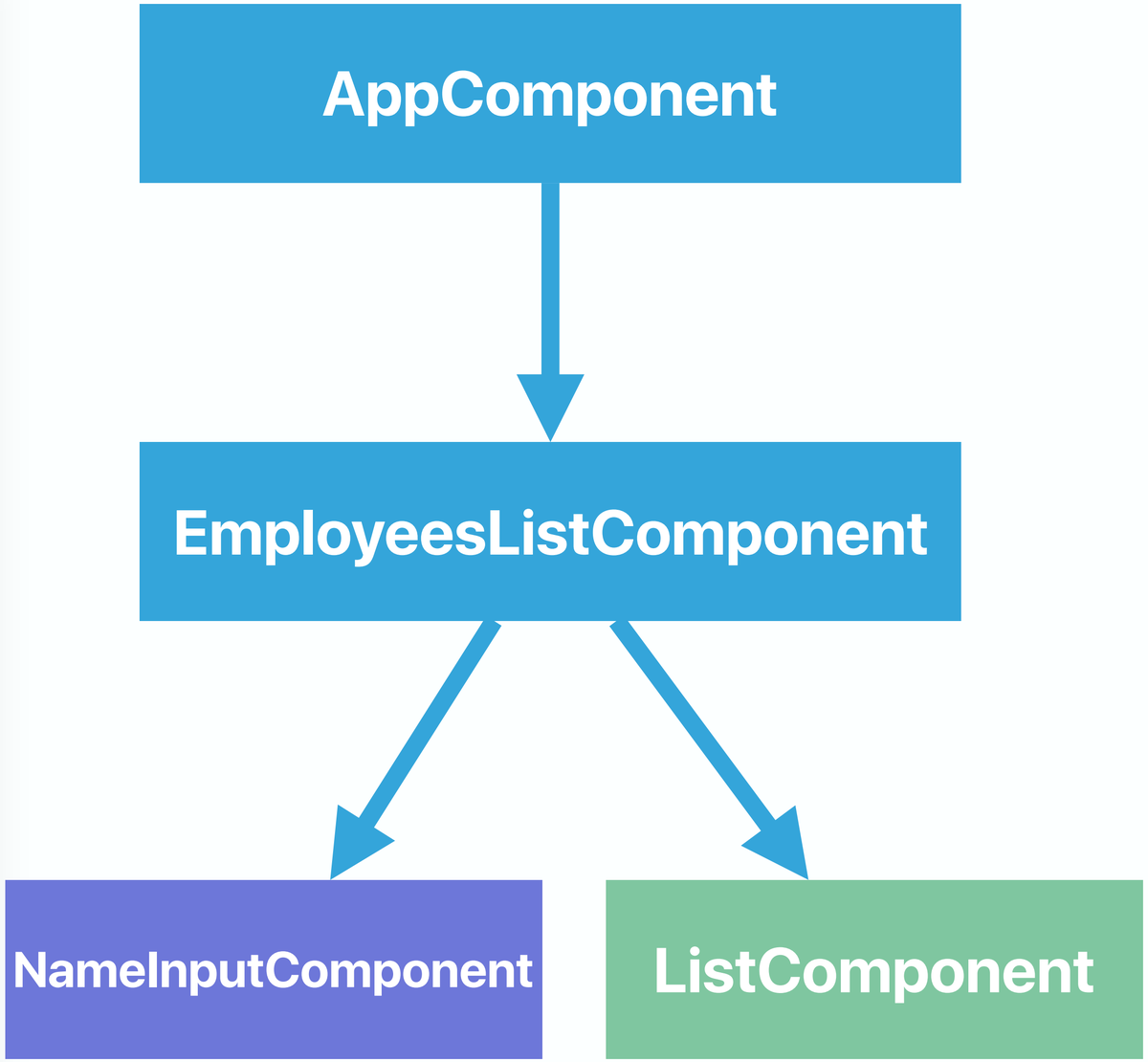
2x faster but still slow...

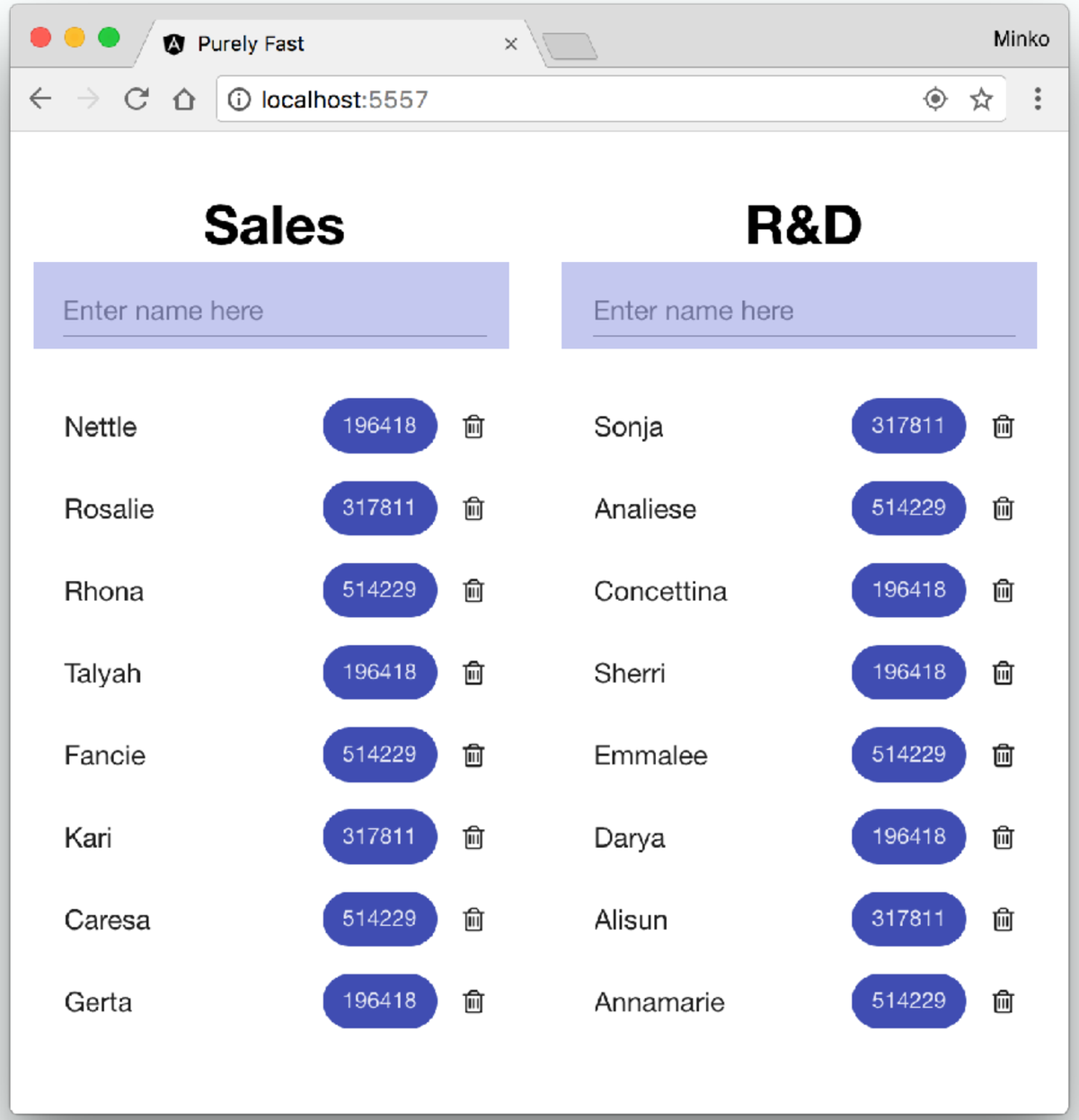

Purely Fast
Minko
localhost:5557
Sales
Enter name here
Nettle 196418
Rosalie 317811
Rhona 514229
Talyah 196418
Fancie 514229
Kari 317811
Caresa 514229
Gerta 196418
Modestine 196418
Elements
Console
top
Filter
Default levels
Console

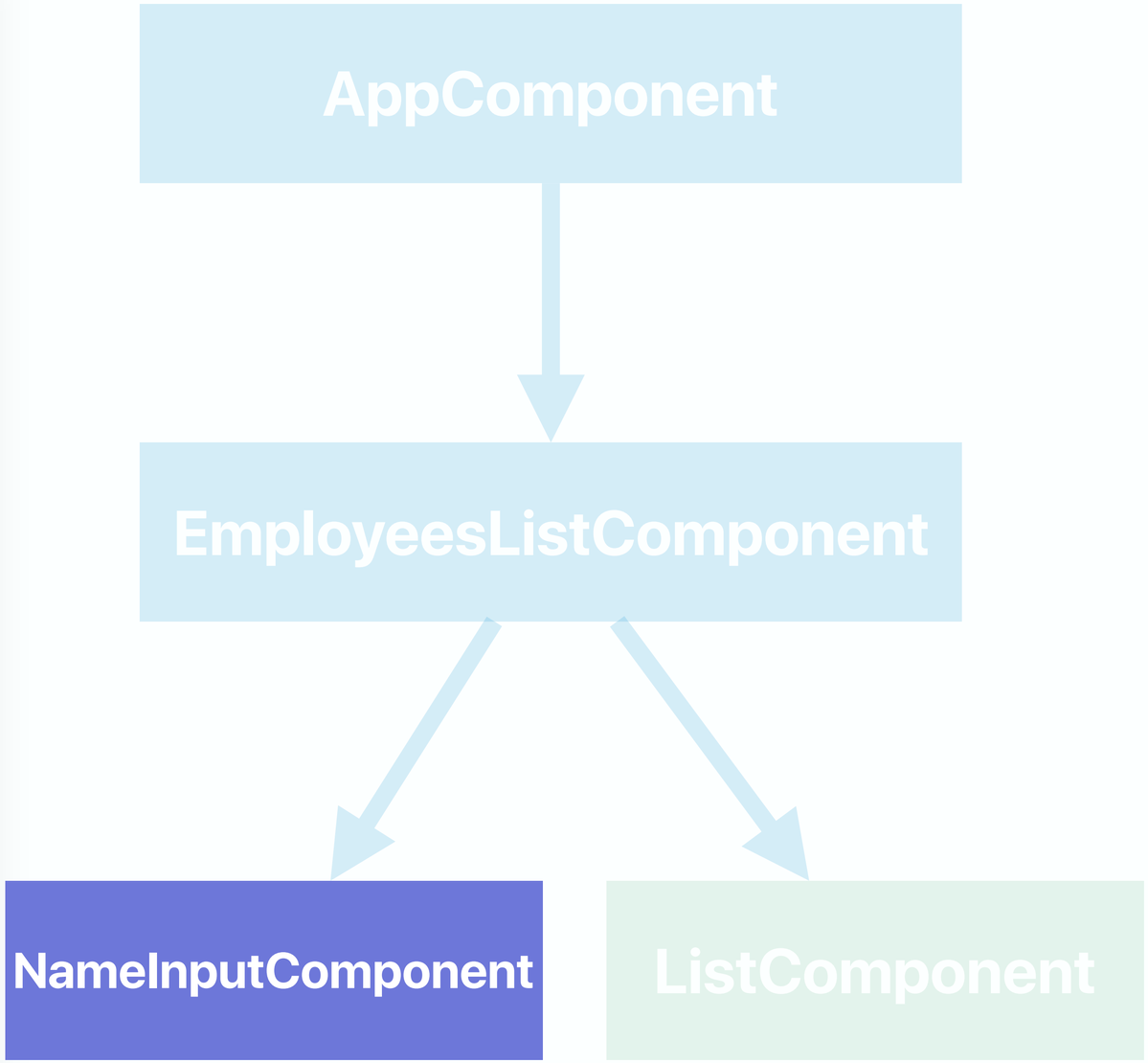
With `OnPush` change detection will be triggered when the framework, with reference check, determines that any of the inputs of a component has changed...**or when an event in the component is triggered**

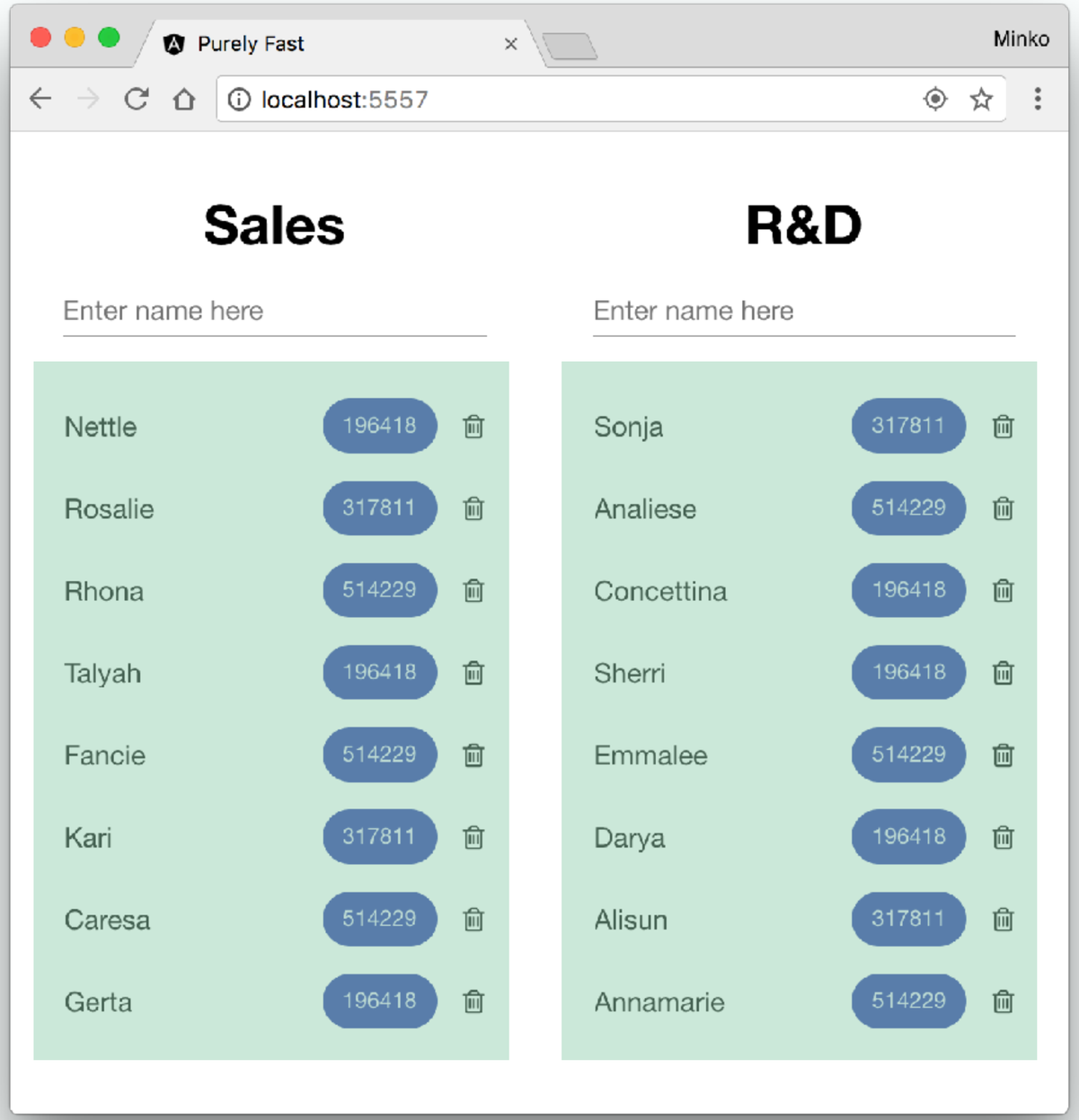
Lets do some refactoring!

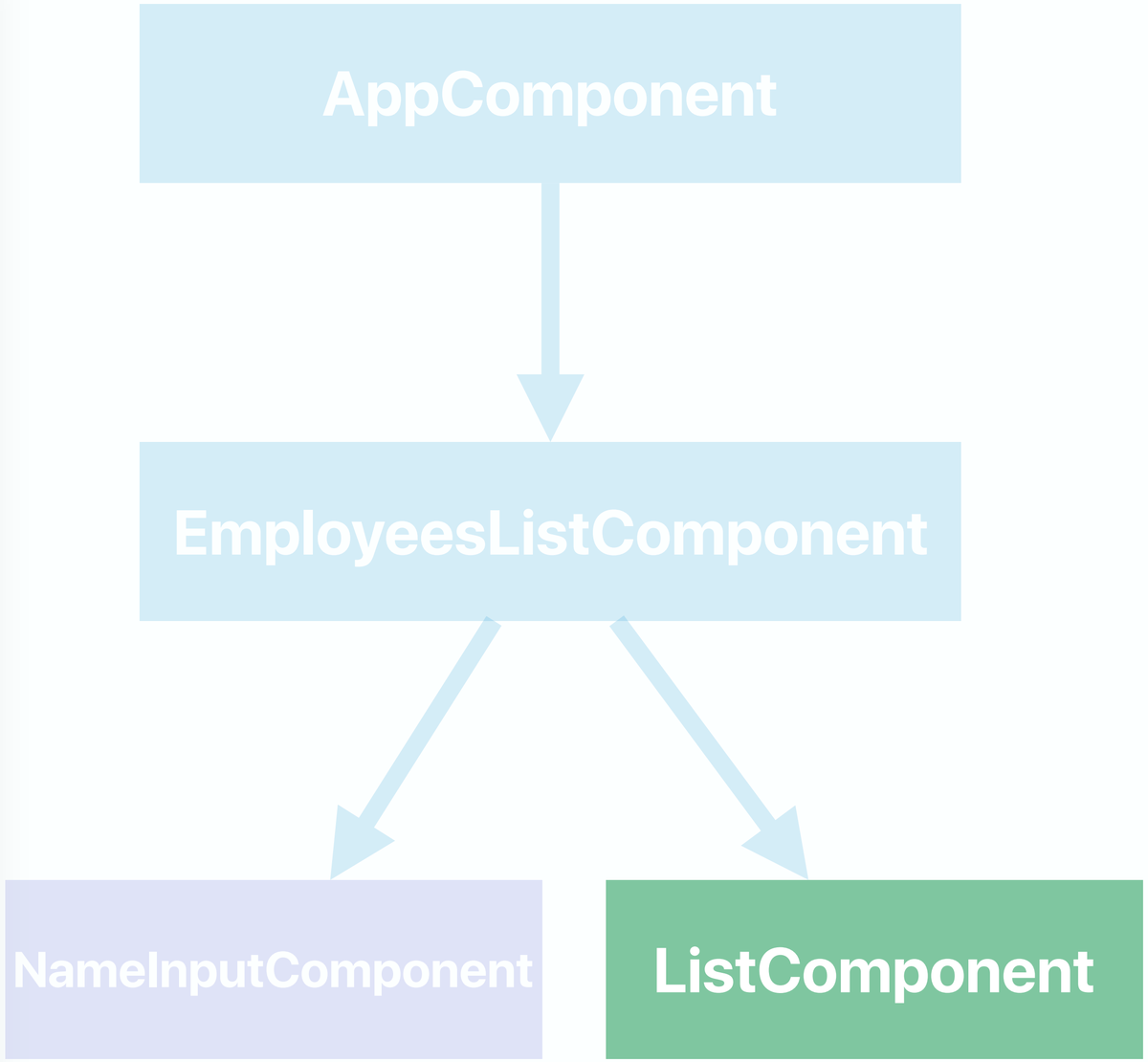

Purely Fast
Minko
localhost:5557
Sales
R&D
Enter name here
Enter name here
Nettle 196418
Rosalie 317811
Rhona 514229
Talyah 196418
Fancie 514229
Kari 317811
Caresa 514229
Gerta 196418
Sonja 317811
Analiese 514229
Concettina 196418
Sherri 196418
Emmalee 514229
Darya 196418
Alisun 317811
Annamarie 514229


AppComponent
EmployeesListComponent
NameInputComponent
ListComponent


Purely Fast
Minko
localhost:5557
Sales
R&D
Enter name here
Enter name here
Nettle 196418
Rosalie 317811
Rhona 514229
Talyah 196418
Fancie 514229
Kari 317811
Caresa 514229
Gerta 196418
Sonja 317811
Analiese 514229
Concettina 196418
Sherri 196418
Emmalee 514229
Darya 196418
Alisun 317811
Annamarie 514229


AppComponent
EmployeesListComponent
NameInputComponent
ListComponent


Purely Fast
Minko
localhost:5557
Sales
R&D
Enter name here
Enter name here
Nettle 196418
Rosalie 317811
Rhona 514229
Talyah 196418
Fancie 514229
Kari 317811
Caresa 514229
Gerta 196418
Sonja 317811
Analiese 514229
Concettina 196418
Sherri 196418
Emmalee 514229
Darya 196418
Alisun 317811
Annamarie 514229


AppComponent
EmployeesListComponent
NameInputComponent
ListComponent

Purely Fast

Minko

localhost:5557

Sales

Enter name here

Nettle	196418
Rosalie	317811
Rhona	514229
Talyah	196418
Fancie	514229
Kari	317811
Caresa	514229
Gerta	196418

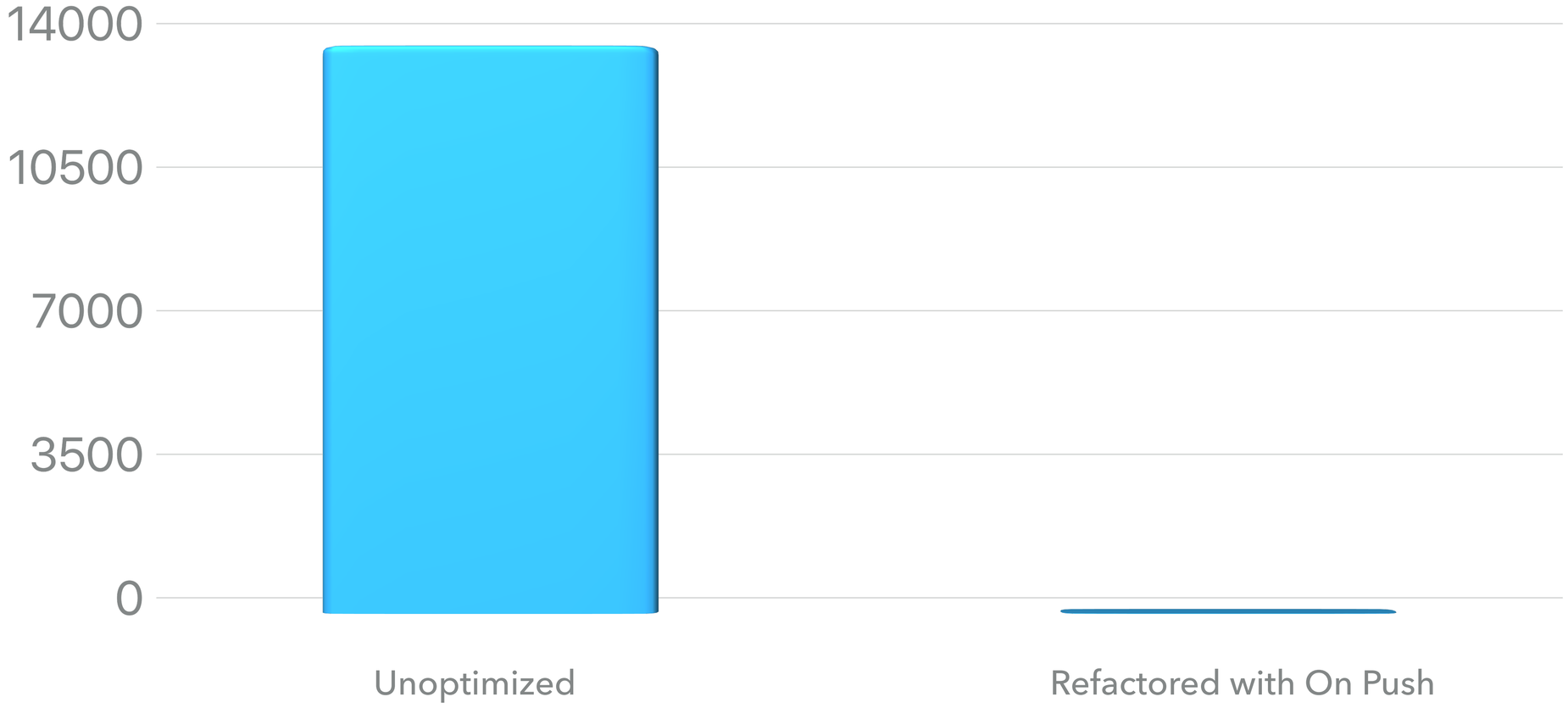
R&D

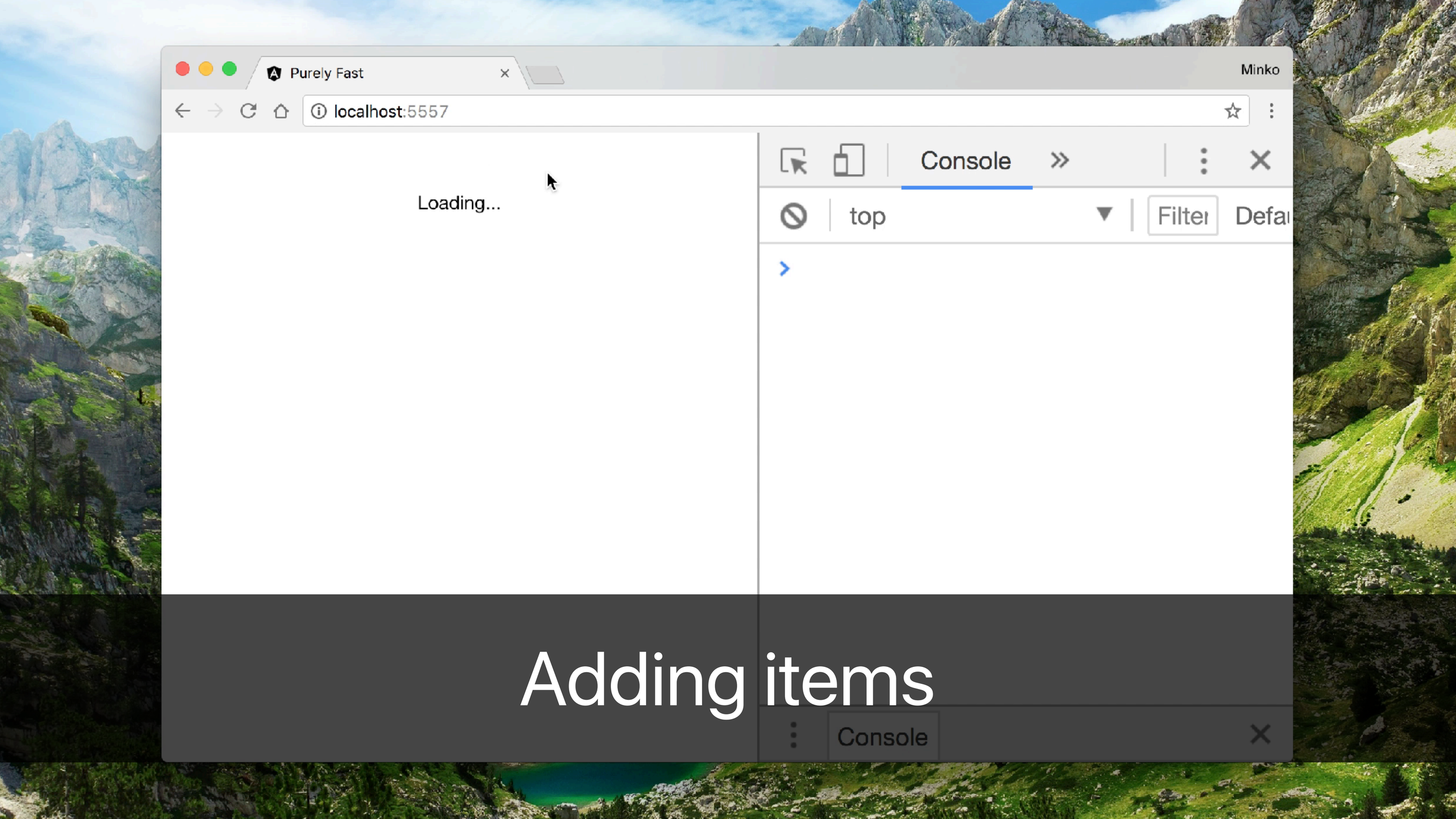
Enter name here

Sonja	317811
Analiese	514229
Concettina	196418
Sherri	196418
Emmalee	514229
Darya	196418
Alisun	317811
Annamarie	514229

Sooo much faster!

Typing Speed





Adding items

Recomputing everything

every time we add a new entry

```
const fibonacci = n => {
  if (n === 1 || n === 2) return 1;
  return fibonacci(n - 1) + fibonacci(n - 2);
};
```

```
const fibonacci = n => {
  if (n === 1 || n === 2) return 1;
  return fibonacci(n - 1) + fibonacci(n - 2);
};


// Two properties
// - No side effects
// - Same result for same arguments
```

Pure Function

Pipes in Angular

- Pure
- Impure

Angular executes a pure pipe only when it detects a change to the input value. A pure change is either a change to a primitive input value (String, Number, Boolean, Symbol) or a changed object reference (Date, Array, Function, Object).

```
@Pipe({ name: 'calculate', pure: true })
export class CalculatePipe {
  transform(num: number) {
    return fibonacci(num);
  }
}
```

```
@Component({
  changeDetection: ChangeDetectionStrategy.OnPush,
  template: `
    ...
     {{ item.num | calculate }}
    ...
  `
})
export class ListComponent { ... }
```

Lets **benchpress** it!

Purely Fast

localhost:5555

Sales

Enter name here

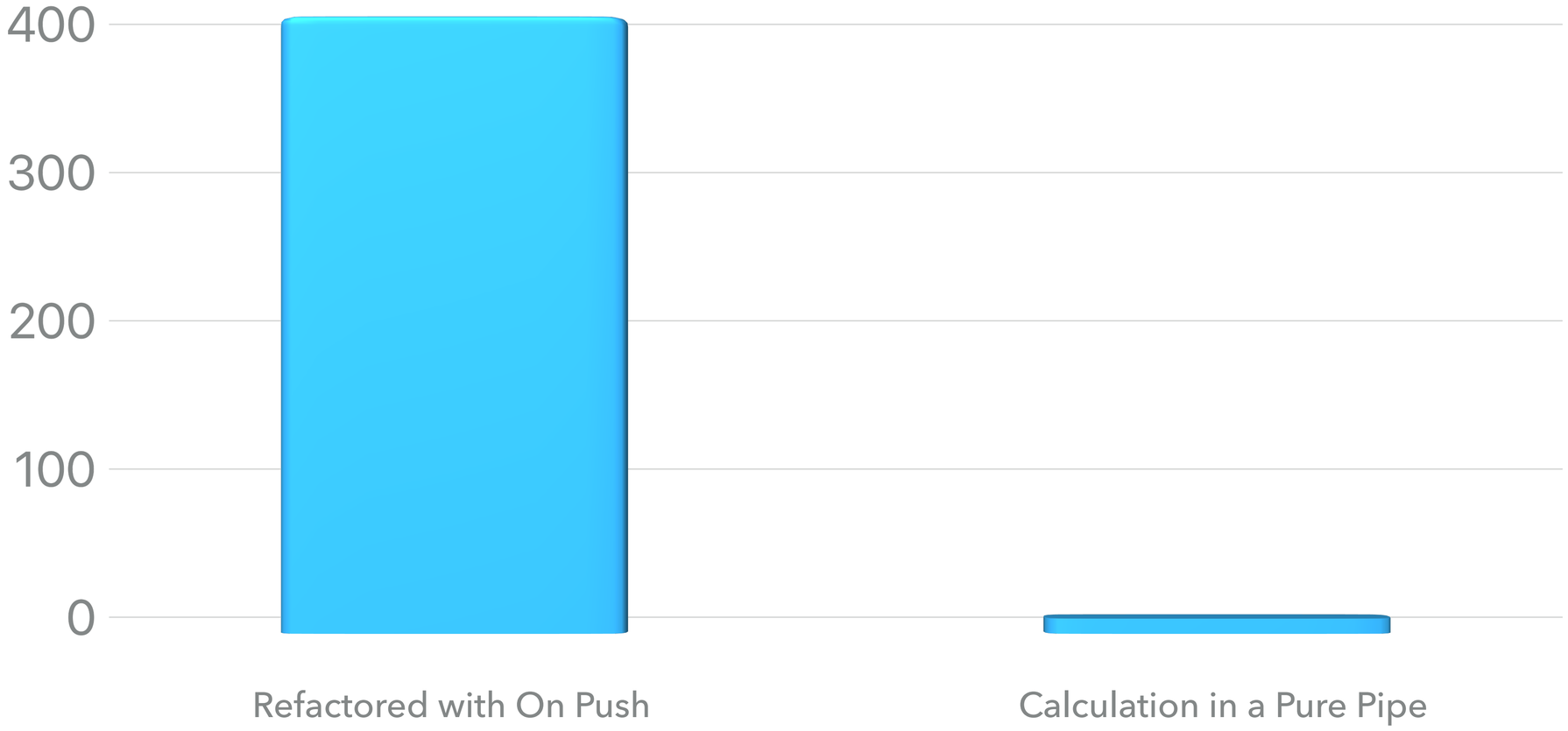
	317811
Rosalie	317811
Rhona	514229
Talyah	196418
Fancie	514229
Kari	317811
Caresa	514229
Gerta	196418

R&D

Enter name here

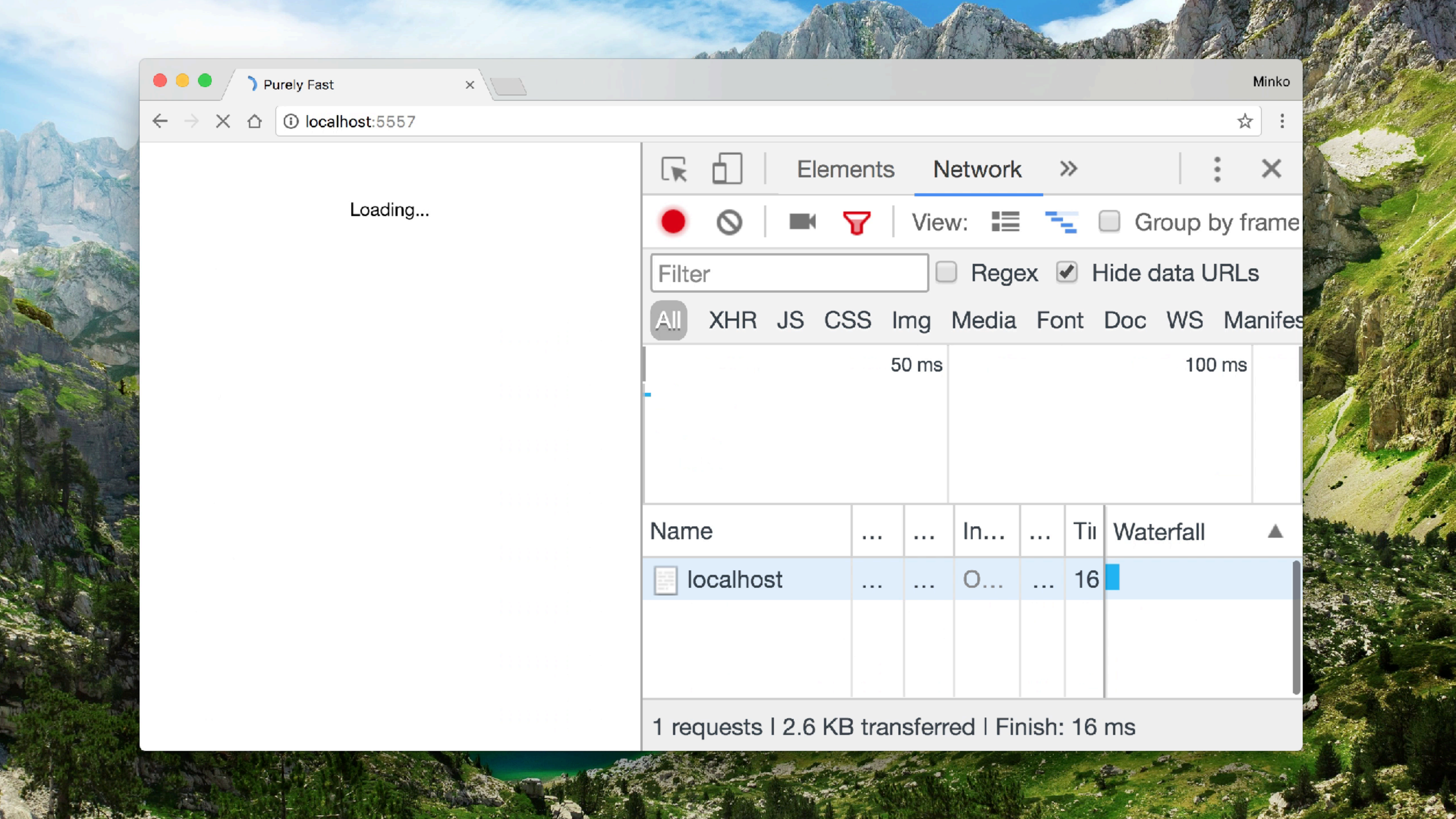
Sonja	317811
Analiese	514229
Concettina	196418
Sherri	196418
Emmalee	514229
Darya	196418
Alisun	317811
Annamarie	514229

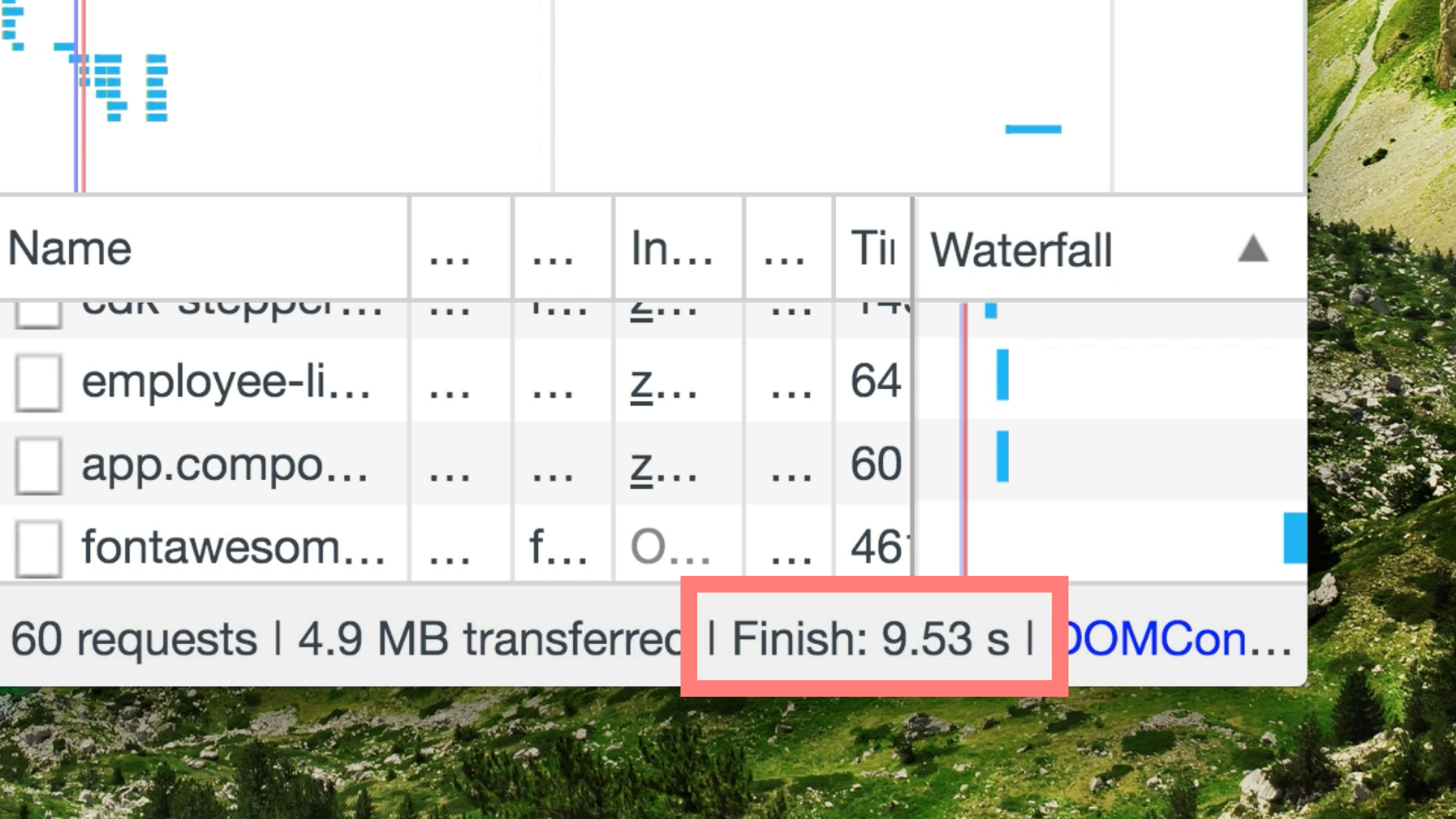
Adding / Removing Entries



Initial rendering...

with 1000 items


Purely Fast
Minko
localhost:5557
Loading...
Elements
Network
View:
Group by frame
Filter
Regex
Hide data URLs
All
XHR
JS
CSS
Img
Media
Font
Doc
WS
Manifes
50 ms
100 ms
Name
Waterfall
localhost
16
1 requests | 2.6 KB transferred | Finish: 16 ms


Name
...
...
In...
...
Ti
Waterfall
employee-li...
...
...
z...
...
64
app.compo...
...
...
z...
...
60
fontawesom...
...
f...
O...
...
46
60 requests | 4.9 MB transferred | Finish: 9.53 s |
DOMCon...

Lets take a look at our data

Sales

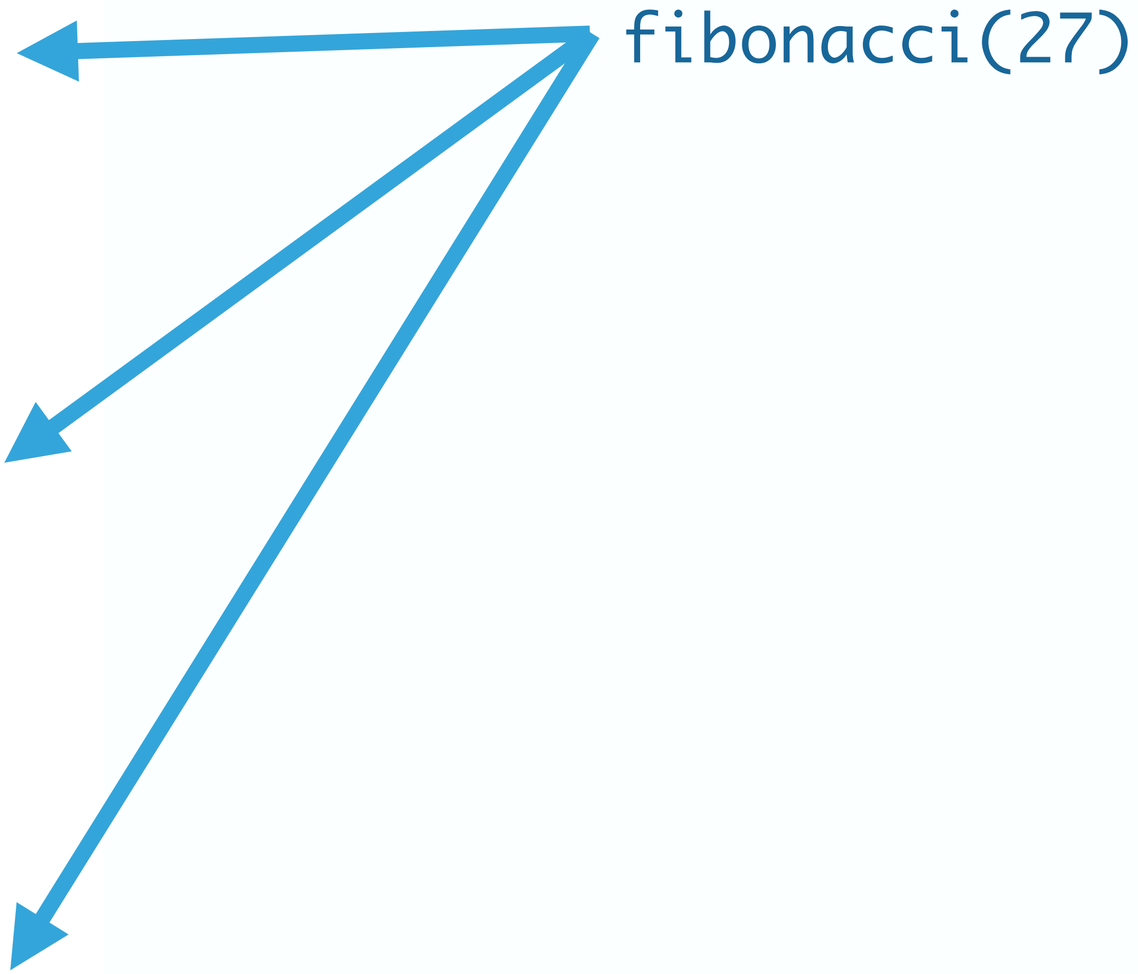
Enter name here

Nettle	196418
Rosalie	317811
Rhona	514229
Talyah	196418
Fancie	514229
Kari	317811
Caresa	514229
Gerta	196418

Sales

Enter name here

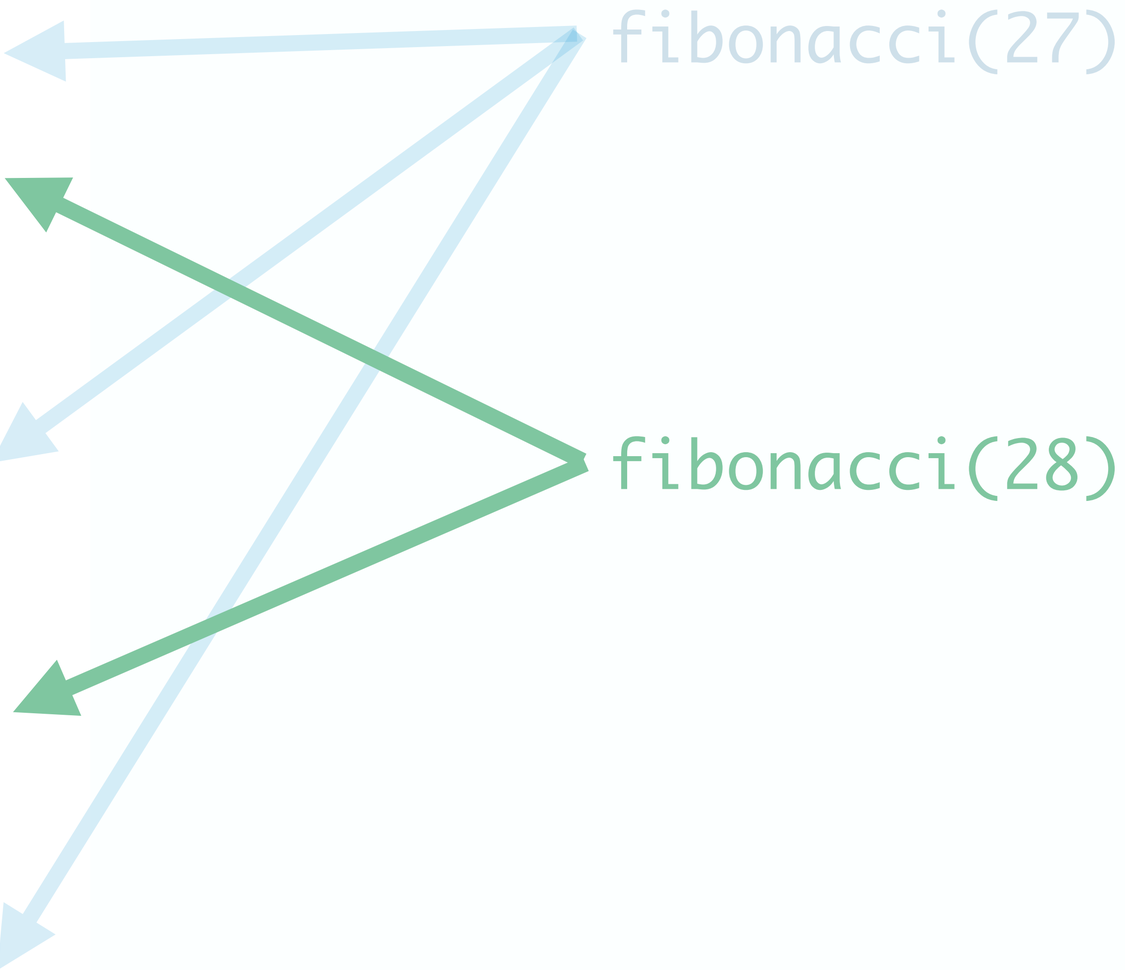
Nettle	196418
Rosalie	317811
Rhona	514229
Talyah	196418
Fancie	514229
Kari	317811
Caresa	514229
Gerta	196418



Sales

Enter name here

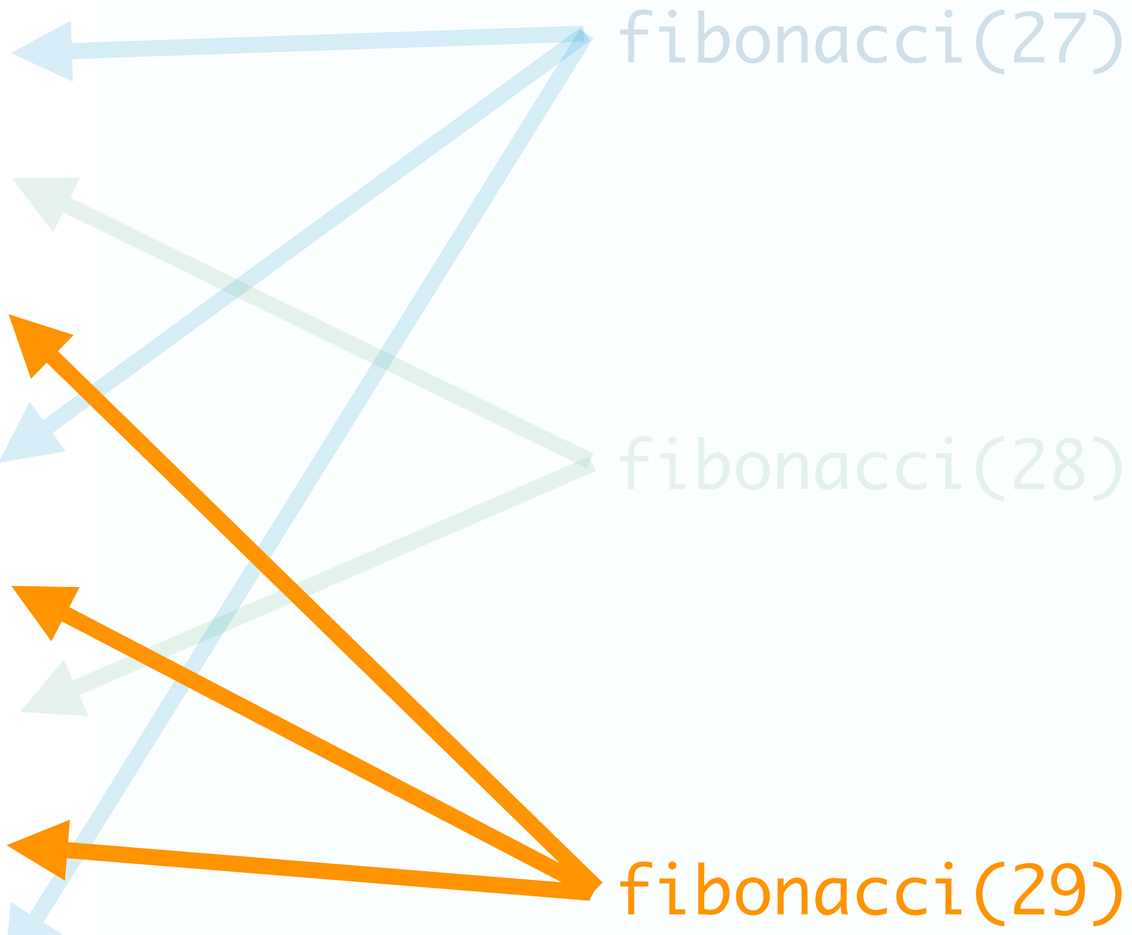
Nettle	196418
Rosalie	317811
Rhona	514229
Talyah	196418
Fancie	514229
Kari	317811
Caresa	514229
Gerta	196418



Sales

Enter name here

Nettle	196418
Rosalie	317811
Rhona	514229
Talyah	196418
Fancie	514229
Kari	317811
Caresa	514229
Gerta	196418



Samples from a small range

During initial rendering we **recompute same value multiple times**

Solution

Caching the value once computed

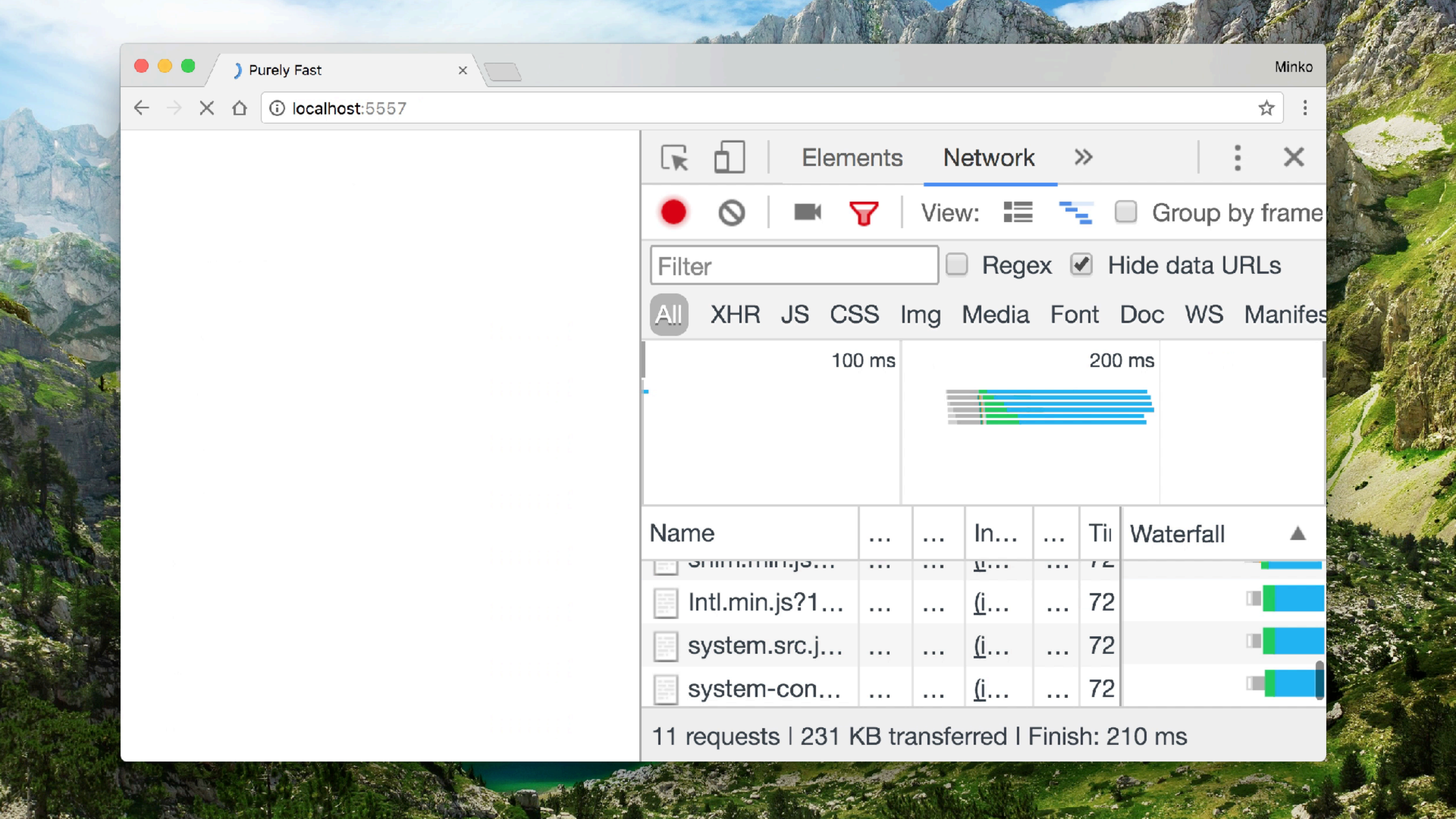
Memoization

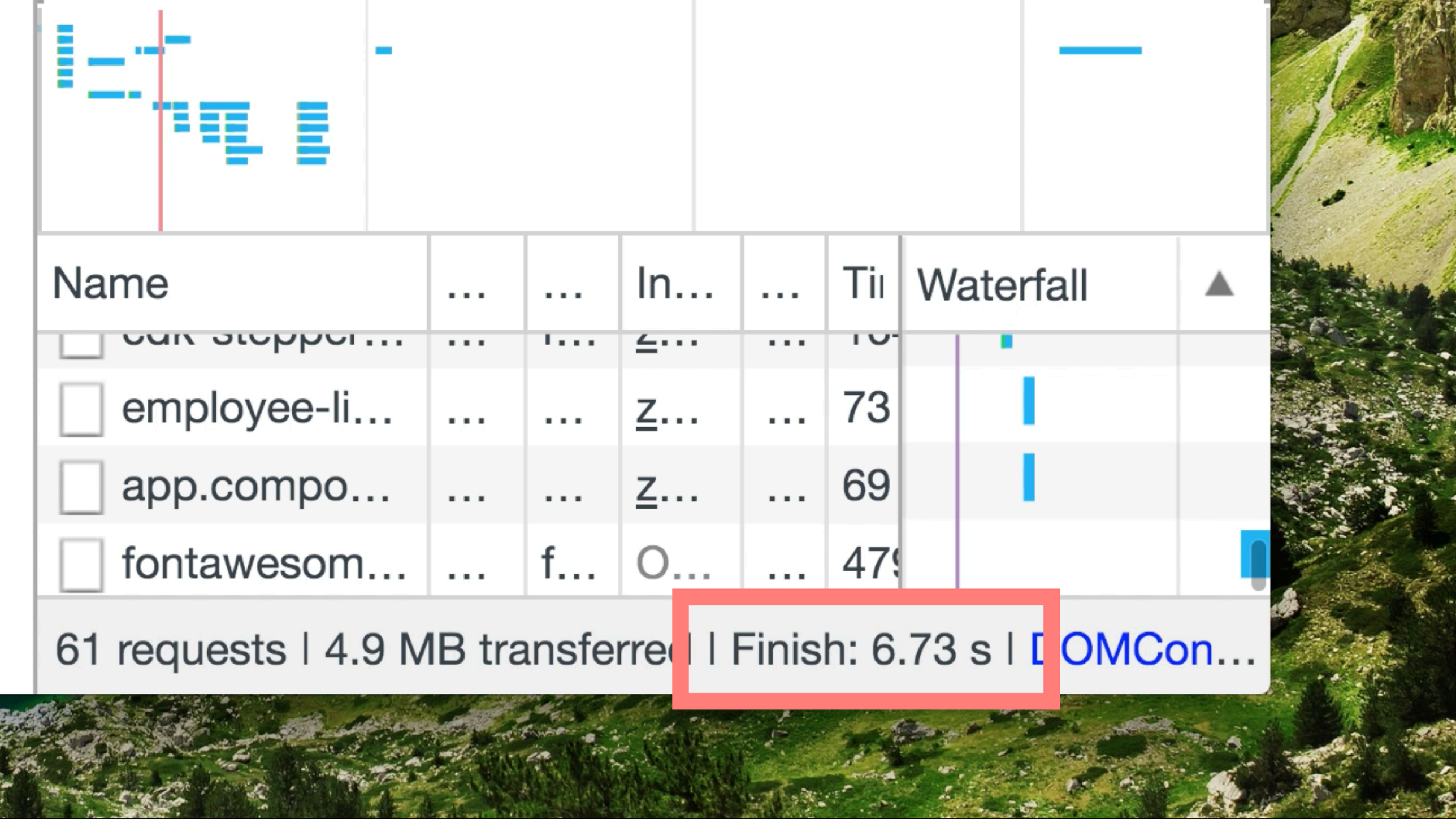
Memoization

possible for pure functions

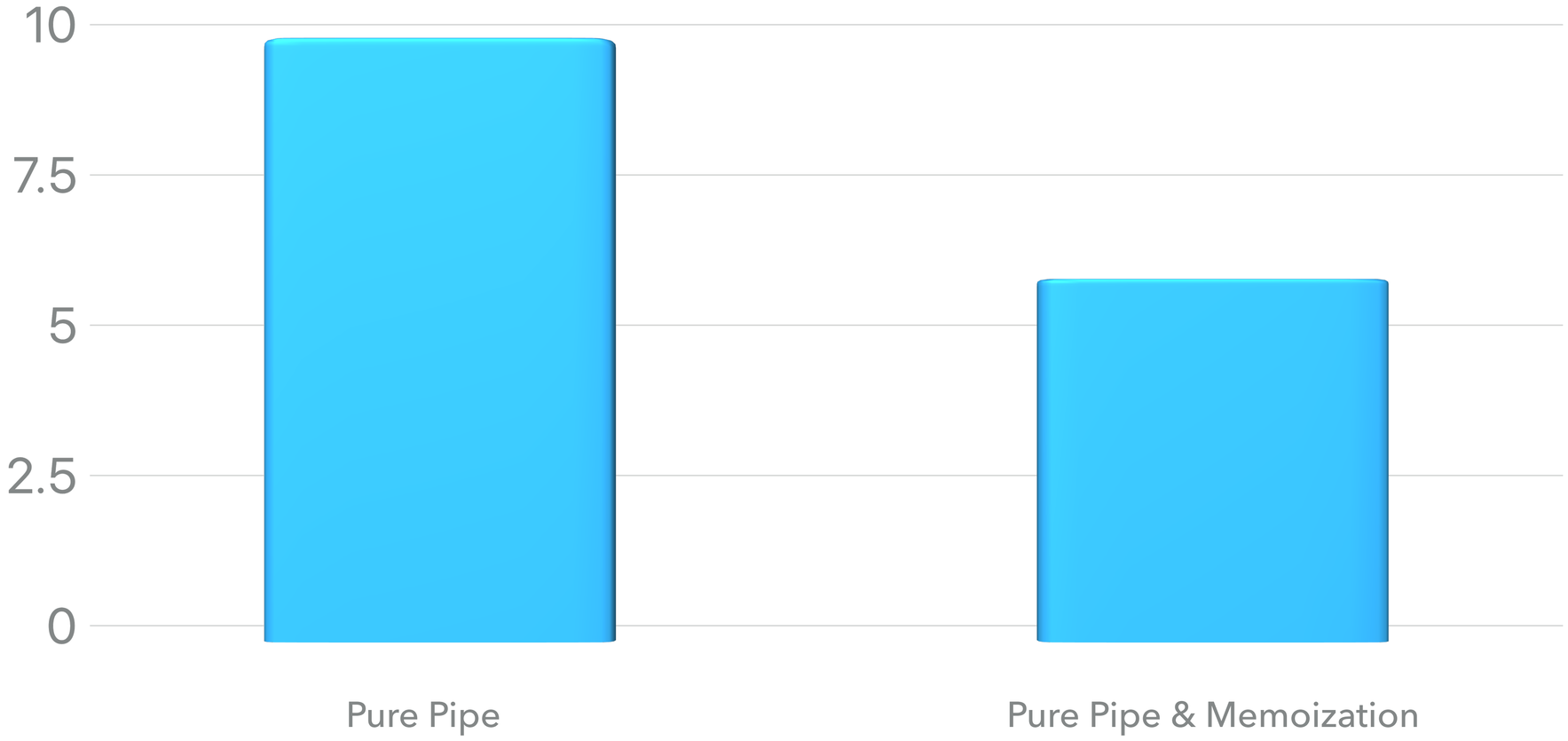
```
const memoize = require('lodash.memoize');

const fibonacci = memoize((num: number): number => {
  if (num === 1 || num === 2) return 1;
  return fibonacci(num - 1) + fibonacci(num - 2);
});
```

```
const memoize = require('lodash.memoize');

const fibonacci = memoize((num: number): number => {
  if (num === 1 || num === 2) return 1;
  return fibonacci(num - 1) + fibonacci(num - 2);
});
```


Purely Fast
Minko
localhost:5557
Elements
Network
View:
Group by frame
Filter
Regex
Hide data URLs
All
XHR
JS
CSS
Img
Media
Font
Doc
WS
Manifes
100 ms
200 ms
Name
Waterfall
Intl.min.js?1…
system.src.j…
system-con…
72
11 requests | 231 KB transferred | Finish: 210 ms


Name
...
...
In...
...
Ti
Waterfall
employee-li...
...
...
z...
...
73
app.compo...
...
...
z...
...
69
fontawesom...
...
f...
O...
...
47
61 requests | 4.9 MB transferre | Finish: 6.73 s | DOMCon...

Initial Rendering



Pure Pipes

Nettle	196418	
Rosalie	317811	
Rhona	514229	
Talyah	196418	
Fancie	514229	
Kari	317811	
Caresa	514229	
Gerta	196418	

```
27 | calculate
```

```
27 | calculate
```

```
27 | calculate
```

Pure Pipes

Nettle	196418
Rosalie	317811
Rhona	514229
Talyah	196418
Fancie	514229
Kari	317811
Caresa	514229
Gerta	196418

```
27 | calculate → fib(27) → 196418
```

```
27 | calculate
```

```
27 | calculate
```

Pure Pipes

Nettle	196418
Rosalie	317811
Rhona	514229
Talyah	196418
Fancie	514229
Kari	317811
Caresa	514229
Gerta	196418

```
27 | calculate → fib(27) → 196418
```

```
27 | calculate → fib(27) → 196418
```

```
27 | calculate
```

Pure Pipes

Nettle	196418	
Rosalie	317811	
Rhona	514229	
Talyah	196418	
Fancie	514229	
Kari	317811	
Caresa	514229	
Gerta	196418	

```
27 | calculate → fib(27) → 196418
```

```
27 | calculate → fib(27) → 196418
```

```
27 | calculate → fib(27) → 196418
```

Memoization

Nettle	196418	`27	calculate`
Rosalie	317811	
Rhona	514229	
Talyah	196418	`27	calculate`
Fancie	514229	
Kari	317811	
Caresa	514229	
Gerta	196418	`27	calculate`

Memoization

Nettle	196418	
Rosalie	317811	
Rhona	514229	
Talyah	196418	
Fancie	514229	
Kari	317811	
Caresa	514229	
Gerta	196418	

```
27 | calculate → fib(27) → 196418
```

```
27 | calculate
```

```
27 | calculate
```

Memoization

Nettle	196418
Rosalie	317811
Rhona	514229
Talyah	196418
Fancie	514229
Kari	317811
Caresa	514229
Gerta	196418

```
27 | calculate → fib(27) → 196418
```

27

cache

```
27 | calculate
```

```
27 | calculate
```

Memoization

Nettle	196418
Rosalie	317811
Rhona	514229
Talyah	196418
Fancie	514229
Kari	317811
Caresa	514229
Gerta	196418

```
27 | calculate → fib(27) → 196418
```

27

27

```
27 | calculate → fib(27)
```

cache

```
27 | calculate
```

Memoization

Nettle	196418
Rosalie	317811
Rhona	514229
Talyah	196418
Fancie	514229
Kari	317811
Caresa	514229
Gerta	196418

```
27 | calculate → fib(27) → 196418
```

27

27

```
27 | calculate → fib(27) → 196418 cache
```

```
27 | calculate
```

Memoization

Nettle	196418
Rosalie	317811
Rhona	514229
Talyah	196418
Fancie	514229
Kari	317811
Caresa	514229
Gerta	196418

```
27 | calculate → fib(27) → 196418
```

27

27

```
27 | calculate → fib(27) → 196418  cache
```

27

```
27 | calculate → fib(27)
```

Memoization

Nettle	196418
Rosalie	317811
Rhona	514229
Talyah	196418
Fancie	514229
Kari	317811
Caresa	514229
Gerta	196418

```
27 | calculate → fib(27) → 196418
                                  27
                                  27
27 | calculate → fib(27) → 196418  cache
                                  27
27 | calculate → fib(27) → 196418
```

Pattern...

- On push performs "memoization"
- Pure pipes are "memoized"

Pattern...

- On push performs "memoization"
- Pure pipes are "memoized"

for their last input

Lets try to do better!

How NgForOf works

```
@Directive({selector: '[ngFor][ngForOf]'})
export class NgForOf<T> implements DoCheck, OnChanges {
  ...

  constructor(private _differs: IterableDiffers) {}

  ngDoCheck(): void {
    const changes = this._differ.diff(this.ngForOf);
    if (changes) this._applyChanges(changes);
  }
  ...
}
```

How NgForOf works

```
@Directive({selector: '[ngFor][ngForOf]'})
export class NgForOf<T> implements DoCheck, OnChanges {
  ...

  constructor(private _differs: IterableDiffers) {}

  ngDoCheck(): void {
    const changes = this._differ.diff(this.ngForOf);
    if (changes) this._applyChanges(changes);
  }
  ...
}
```

How NgForOf works

```
@Directive({selector: '[ngFor][ngForOf]'})
export class NgForOf<T> implements DoCheck, OnChanges {
  ...

  constructor(private _differs: IterableDiffers) {}

  ngDoCheck(): void {
    const changes = this._differ.diff(this.ngForOf);
    if (changes) this._applyChanges(changes);
  }
  ...
}
```

IterableDiffer checks whether the data structure has changed

But the data structure knows that best!

```
export class DifferableList<T> {
  changes = new LinkedList<IterableChangeRecord<T>>();

  constructor(private data = List<T>([])) {}

  unshift(data: T) {
    const result = new DifferableList<T>(this.data.unshift(data));
    result.changes.add({ ... });
    return result;
  }

  ...

  [Symbol.iterator]() {
    return new DifferableListIterator<T>(this);
  }
}
```

```
export class DifferableList<T> {
  changes = new LinkedList<IterableChangeRecord<T>>();

  constructor(private data = List<T>([])) {}

  unshift(data: T) {
    const result = new DifferableList<T>(this.data.unshift(data));
    result.changes.add({ ... });
    return result;
  }

  ...

  [Symbol.iterator]() {
    return new DifferableListIterator<T>(this);
  }
}
```

```
export class DifferableList<T> {
  changes = new LinkedList<IterableChangeRecord<T>>();

  constructor(private data = List<T>([])) {}

  unshift(data: T) {
    const result = new DifferableList<T>(this.data.unshift(data));
    result.changes.add({ ... });
    return result;
  }


  ...

  [Symbol.iterator]() {
    return new DifferableListIterator<T>(this);
  }
}
```

```
export class DifferableList<T> {
  changes = new LinkedList<IterableChangeRecord<T>>();

  constructor(private data = List<T>([])) {}

  unshift(data: T) {
    const result = new DifferableList<T>(this.data.unshift(data));
    result.changes.add({ ... });
    return result;
  }

  ...

  [Symbol.iterator]() {
    return new DifferableListIterator<T>(this);
  }
}
```

```
export class DifferableList<T> {
  changes = new LinkedList<IterableChangeRecord<T>>();

  constructor(private data = List<T>([])) {}

  unshift(data: T) {
    const result = new DifferableList<T>(this.data.unshift(data));
    result.changes.add({ ... });
    return result;
  }

  ...

  [Symbol.iterator]() {
    return new DifferableListIterator<T>(this);
  }
}
```

Data structure optimized for Angular

```
export class DifferableListDiffer<V>
  implements IterableDiffer<V>, IterableChanges<V> {
    ...

    diff(collection: NgIterable<V>): DifferableListDiffer<V> | null {
      const changes = this._data.changes;
      this._changes = changes;
      if (changes.size() > 0) {
        this._data.changes = new LinkedList<IterableChangeRecord<V>>();
        return this;
      } else {
        return null;
      }
    }
  }
```

```
export class DifferableListDiffer<V>
  implements IterableDiffer<V>, IterableChanges<V> {
    ...

    diff(collection: NgIterable<V>): DifferableListDiffer<V> | null {
      const changes = this._data.changes;
      this._changes = changes;
      if (changes.size() > 0) {
        this._data.changes = new LinkedList<IterableChangeRecord<V>>();
        return this;
      } else {
        return null;
      }
    }
  }
```

Required refactoring

```
@Component({
  selector: 'sd-employee-list',
  providers: [
    IterableDiffers.extend([
      new DifferableListDifferFactory()
    ])
  ],
  template: `...`,
})
export class EmployeeListComponent { ... }
```

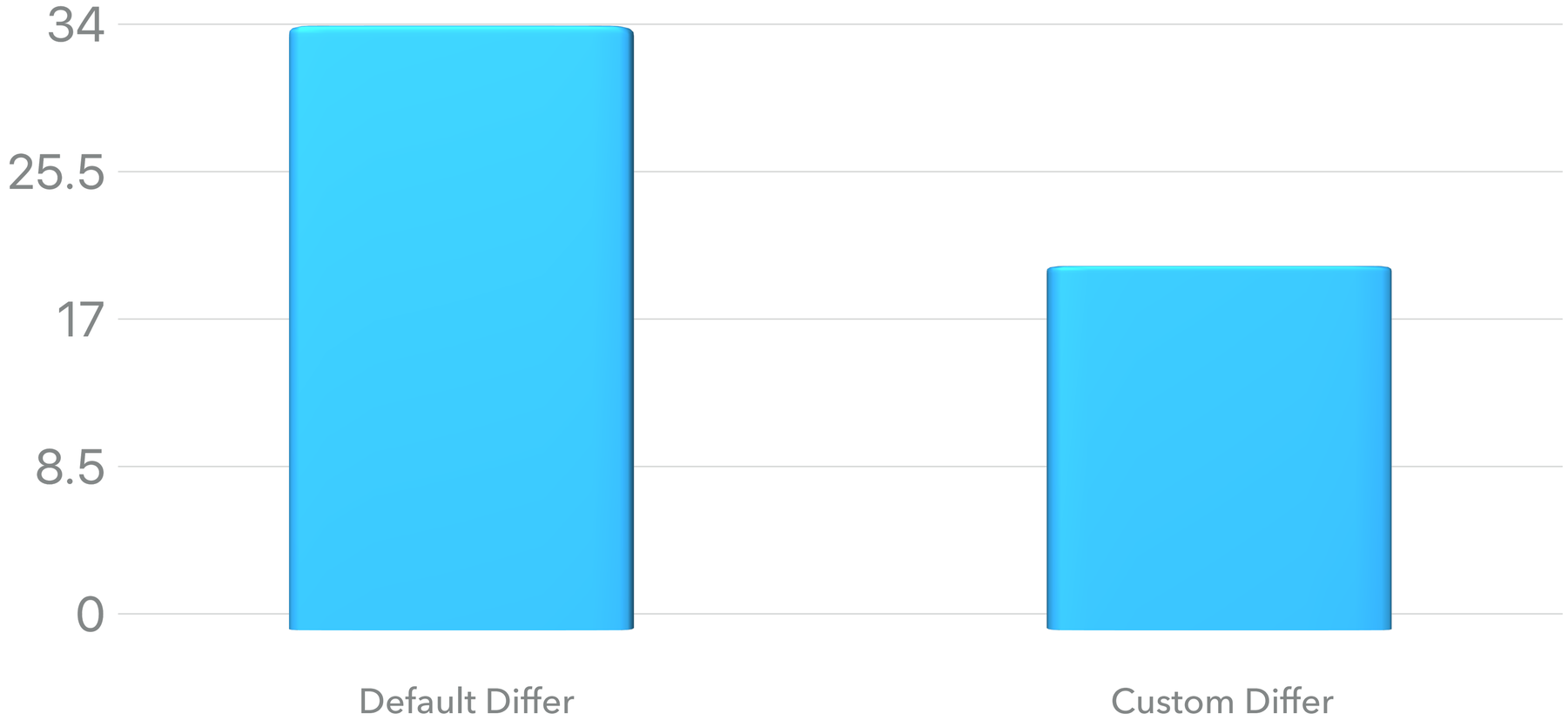
Inspired by

Persistent

data structures

twitter.com/mgechev

Adding / Removing Entries



Lessons learned

- No silver bullet
- Understand your component structure
- Understand your data
- Application specific benchmarks

Get inspiration from **computer science**

twitter.com/mgechev

Links

mgv.io/ng-cd – Angular's OnPush Change Detection Strategy

mgv.io/ng-pure – Pure Pipes and Referential Transparency

mgv.io/ng-diff – Understanding Angular Differs

mgv.io/ng-perf-checklist – Angular Performance Checklist

Thank you!

twitter.com/mgechev

github.com/mgechev

blog.mgechev.com