```
const source = new Observable((observer) => {
  const socket = new WebSocket('ws://someurl');
  socket.addEventListener('message', (e) => observer.next(e));
  return () => socket.close();
});
```

```
const socket = new WebSocket('ws://someurl');

const source = new Observable((observer) => {
 socket.addEventListener('message', (e) => observer.next(e));
});
```

Tip:
Pensate alla definizione di Ben Lesh come una conseguenza, non come punto di partenza. All'Observer non interessa del produttore, e pensarla in questi termini può trarvi in inganno.
Ricordate però che un Observable COLD genera nuovi valori per ogni subscription, anche se possono sembrare gli stessi.

```
const source = new Observable((observer) => {
  const socket = new WebSocket('ws://someurl');
  socket.addEventListener('message', (e) => observer.next(e));
  return () => socket.close();
});
```

```
const socket = new WebSocket('ws://someurl');

const source = new Observable((observer) => {
 socket.addEventListener('message', (e) => observer.next(e));
});
```

Tip:
Pensate alla definizione di Ben Lesh come una conseguenza, non come punto di partenza. All'Observer non interessa del produttore, e pensarla in questi termini può trarvi in inganno.
Ricordate però che un Observable COLD genera nuovi valori per ogni subscription, anche se possono sembrare gli stessi.

Rendiamo HOT un Observable COLD

Introduciamo publish e connect

LIVE CODING