

Asynchronism

Willian Z

Recap

```
function take_lonnnnnng_time(){  
    visit_google()  
    go_get_a_pizza_from_pizza_man()  
}
```

```
async function take_lonnnnnng_time(){  
    visit_google()  
    go_get_a_pizza_from_pizza_man()  
}
```

```
async function take_lonnnnnng_time(){...}
```

```
async function take_lonnnnnng_time(){...}
```

```
take_lonnnnnng_time()
```

```
async function take_lonnnnnng_time(){...}
```



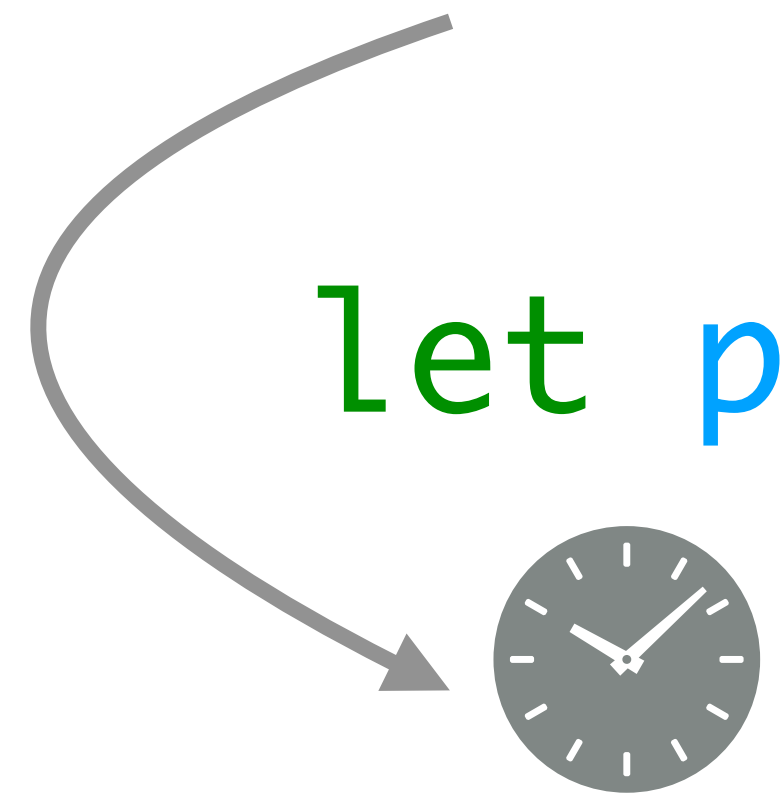
```
async function take_1onnnnnng_time(){...}
```

`await` take_1onnnnnng_time()



```
async function take_lonnnnnng_time(){  
    visit_google()  
    return go_get_a_pizza_from_pizza_man()  
}
```

`let pizza = await take_lonnnnnng_time()`



```
async function take_lonnnnnng_time(){  
    visit_google()  
    return go_get_a_pizza_from_pizza_man()  
}
```

let pizza = await take_lonnnnnng_time()



take_lonnnnnng_time()

Instantly!

```
async function take_lonnnnnng_time(){...}
```

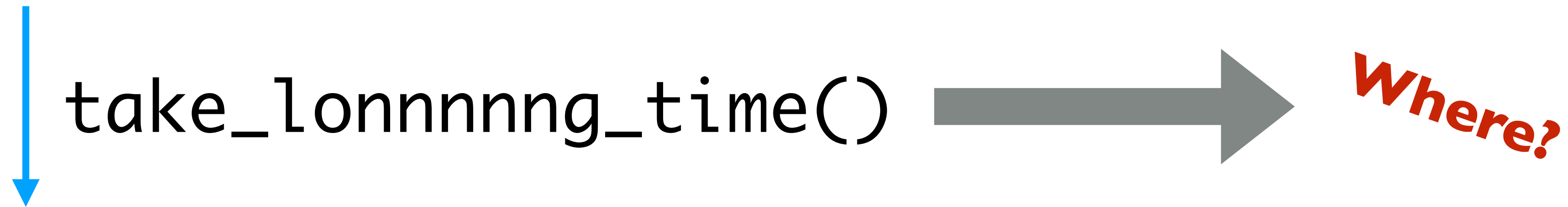
```
let pizza = await take_lonnnnnng_time()
```



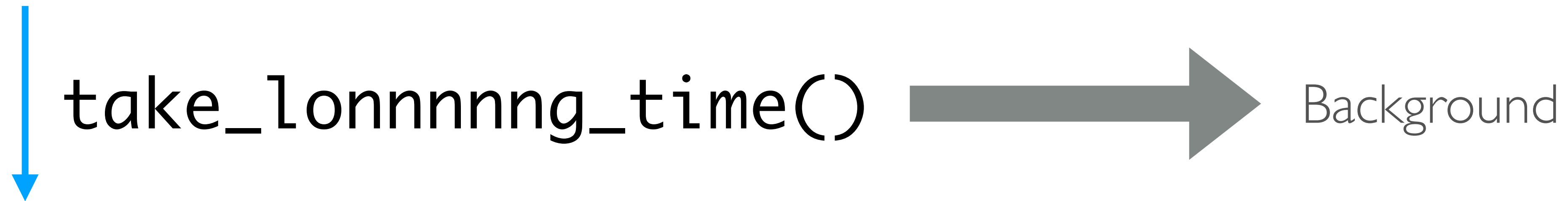
```
take_lonnnnnng_time()
```

Instantly!

`async function take_lonnnnnng_time(){...}`



```
async function take_lonnngng_time(){...}
```



Topics

- Parallelism with javascript async/await
- Efficient Parallelism

Topics

- Parallelism with javascript async/await
- Efficient Parallelism

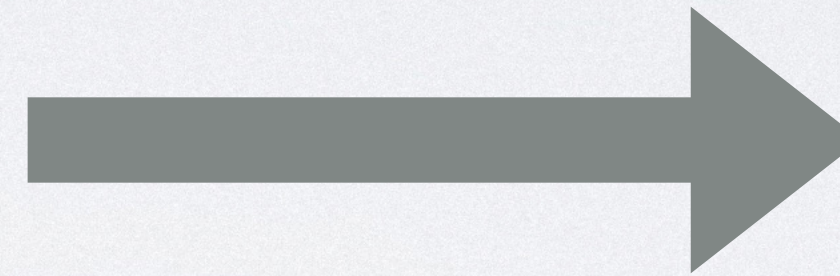
```
async function make_a_pizza(){...}
```



`async function make_a_pizza(){...}`



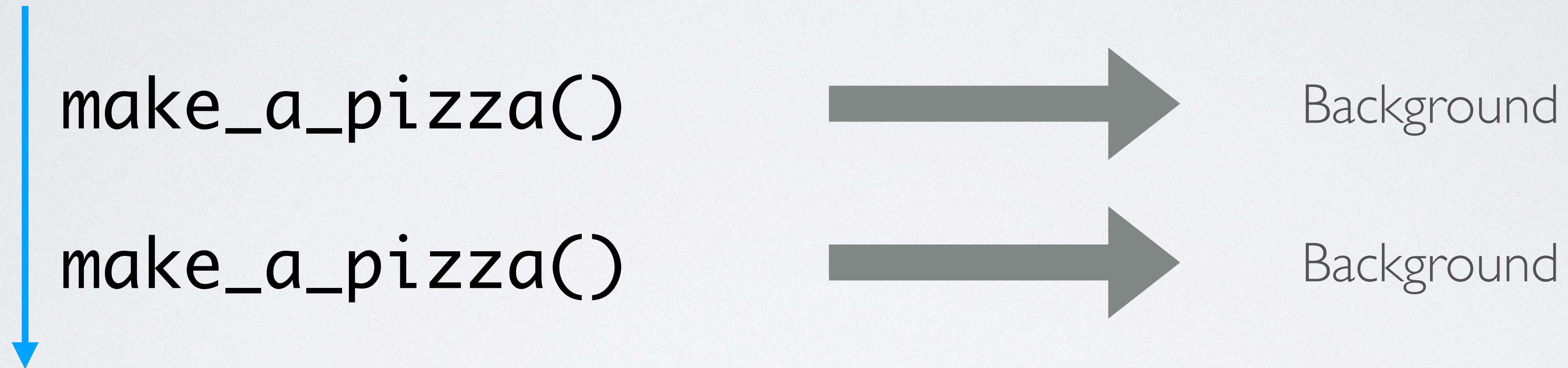
`make_a_pizza()`



Background



`async function make_a_pizza(){...}`



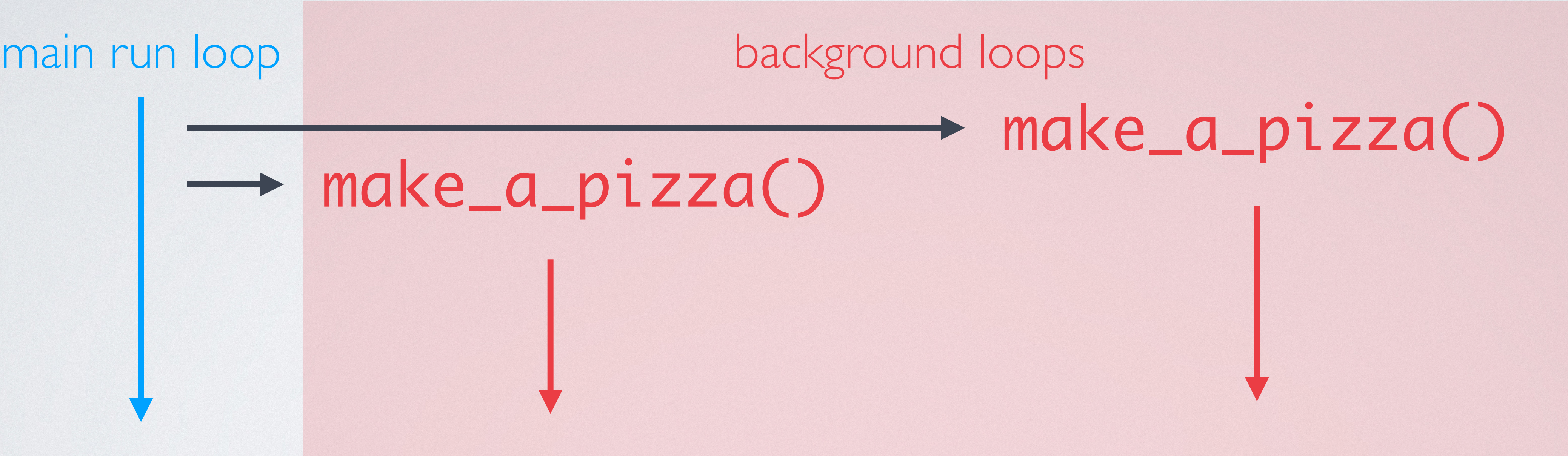


`async function make_a_pizza(){...}`





```
async function make_a_pizza(){...}
```



JS

`async function make_a_pizza(){...}`

main run loop

background loops

`make_a_pizza()` → `make_a_pizza()`



Where are the Pizzas?

```
async function make_a_pizza(){...}
```

```
async function make_a_pizza(){...}
```

=

Promise

Recap
Promise

```
new Promise ( )
```

```
new Promise().then()
```

```
new Promise().then().then()
```

```
new Promise().then().then().then()
```

JS

```
new Promise().then(  
  (result)=>{  
    return something  
  }  
).then().then()
```

JS

```
async function make_a_pizza(){...}
```

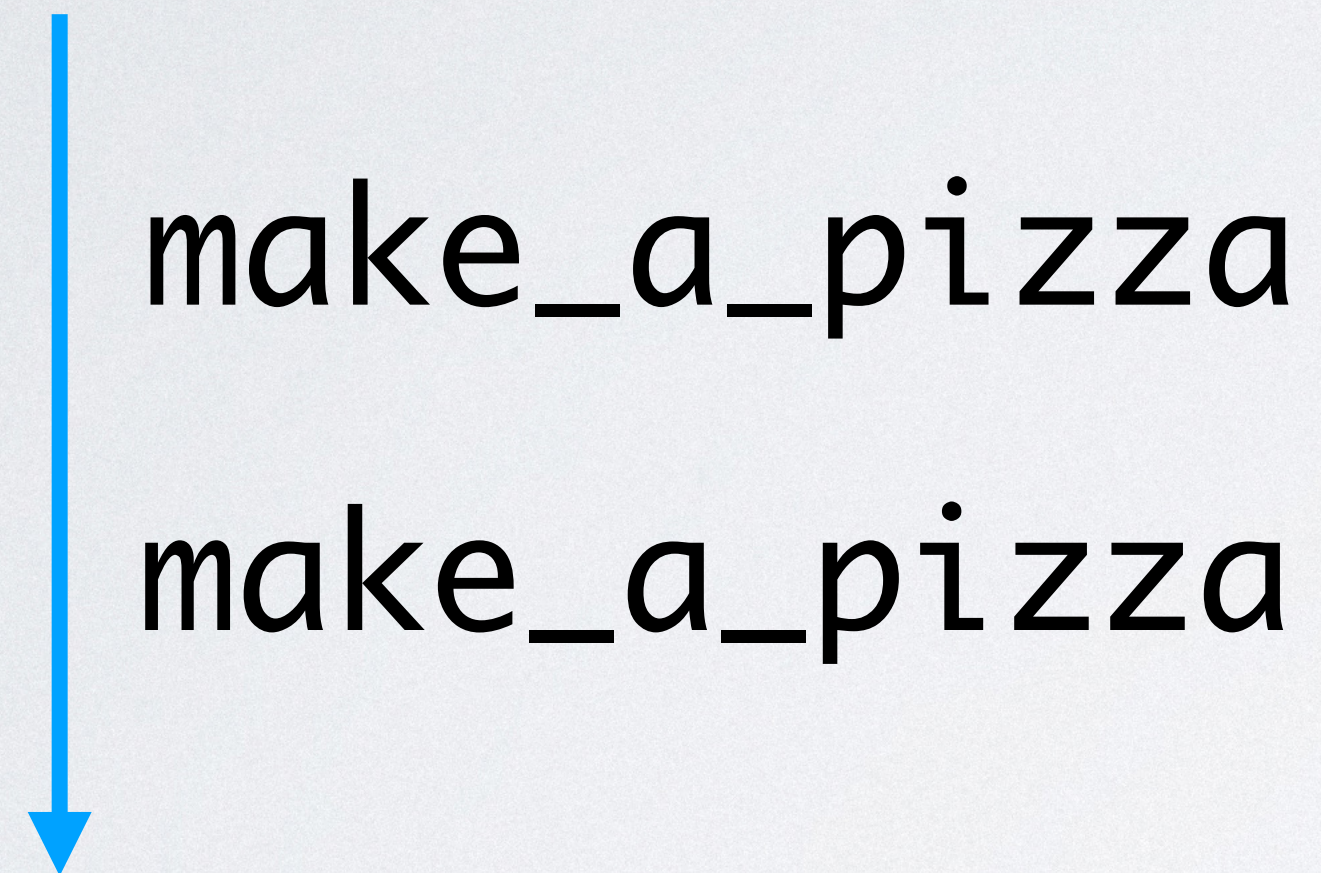
```
make_a_pizza().then(
```

```
  (pizza)=>{  
    return something  
  }
```

```
).then(  ).then(  )
```

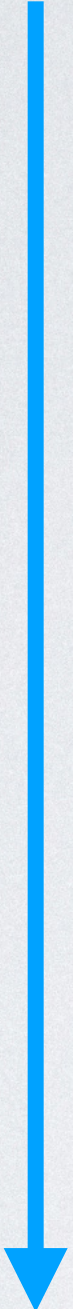
How About Two Running in Parallel?

```
async function make_a_pizza() {...}
```



JS

`async function make_a_pizza() {...}`



```
Promise.all([  
  make_a_pizza,  
  make_a_pizza  
])
```

JS

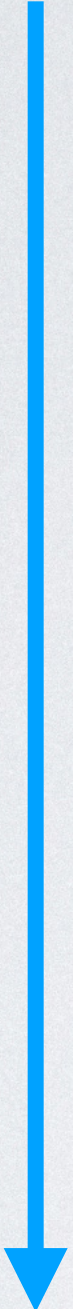
async function make_a_pizza() {...}



```
Promise.all([  
  make_a_pizza,  
  make_a_pizza  
]).then(  
  (pizzas) => {  
    ...  
  }  
)
```

JS

`async function make_a_pizza() {...}`



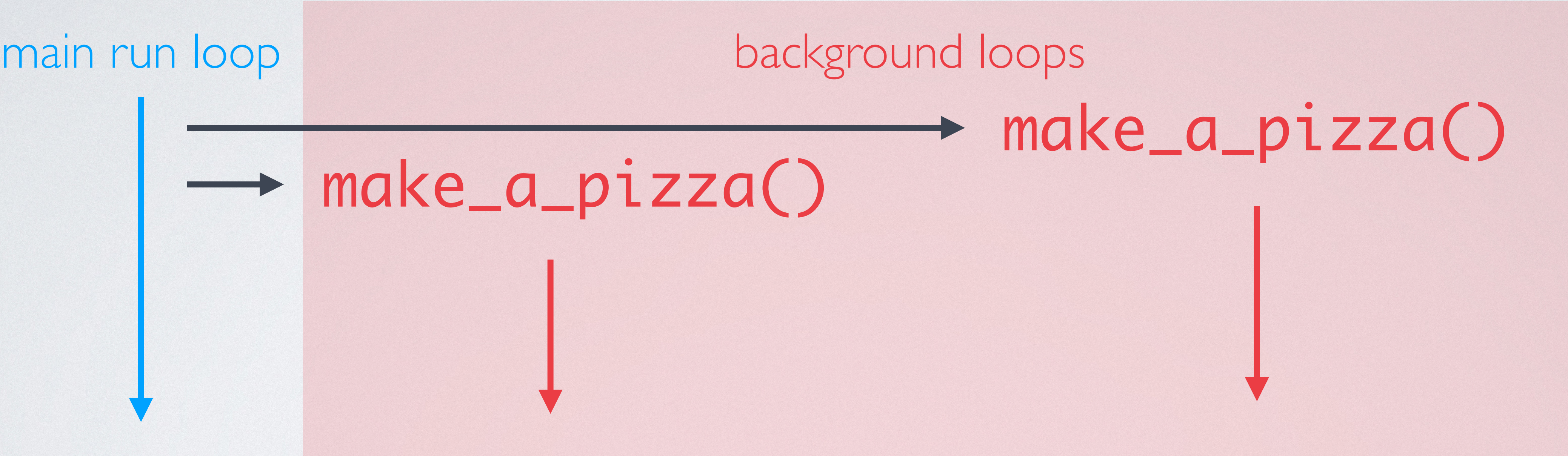
```
let pizzas = await Promise.all([  
  make_a_pizza,  
  make_a_pizza  
])
```

Topics

- Parallelism with javascript async/await
- Efficient Parallelism



```
async function make_a_pizza(){...}
```

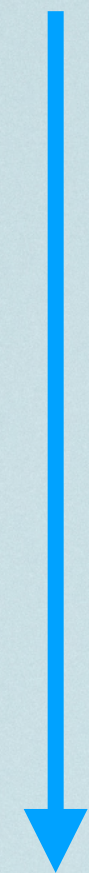


JS

`async function make_a_pizza(){...}`

main run loop

background loops



`make_a_pizza()`

`make_a_pizza()`



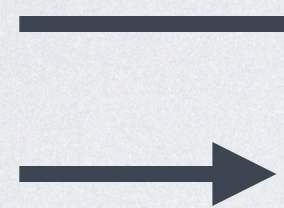
One System Thread



`async function make_a_pizza(){...}`

main run loop

background loops



`make_a_pizza()`



`make_a_pizza()`





`async function make_a_pizza(){...}`

main thread

background threads



`make_a_pizza()`

`make_a_pizza()`





Swift Recap



```
func make_a_pizza() {  
  
}
```



```
func make_a_pizza() -> Pizza {  
  
}
```



```
async func make_a_pizza() {  
  
}
```



```
async func make_a_pizza() {  
  
}
```

*Not available **yet**



```
async {  
  
    make_a_pizza()  
  
}
```

<https://github.com/yannickl/AwaitKit>



```
async {  
    make_a_pizza()  
}
```



```
async {  
    await ( make_a_pizza() )  
}
```



```
func make_a_pizza() -> Promise<Pizza>{ ... }
```

```
async {
```

```
    await( make_a_pizza() )
```

```
}
```



```
func make_a_pizza() -> Promise<Pizza>{ ... }
```

```
async {
```

```
    await( make_a_pizza() )
```

```
}
```

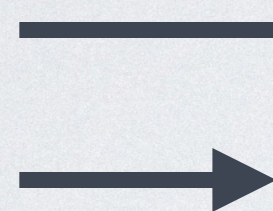
That's It



```
func make_a_pizza() -> Promise<Pizza>{ ... }
```

main thread

background threads



`make_a_pizza()`



`make_a_pizza()`

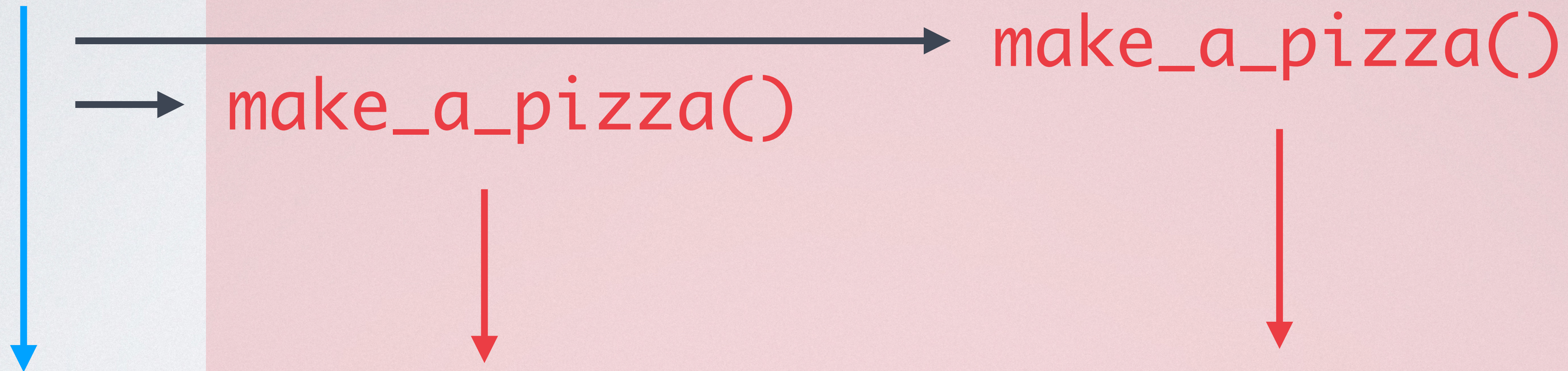




```
func make_a_pizza() -> Promise<Pizza>{ ... }
```

main thread

background threads



Python except you have to manage Context explicitly



```
import asyncio
```

Python except you have to manage Context explicitly



```
async def make_a_pizza()
```

Python except you have to manage Context explicitly



```
async def make_a_pizza()
```

```
tasks = [  
    asyncio.ensure_future( make_a_pizza() )  
    for _ in range(total_pizzas_you_want)  
]
```

Python except you have to manage Context explicitly



```
async def make_a_pizza()  
    tasks = [...]
```

```
loop = asyncio.get_event_loop()
```

Python except you have to manage Context explicitly



```
async def make_a_pizza()  
tasks = [...]  
loop = asyncio.get_event_loop()
```

```
loop.run_until_complete(asyncio.wait(tasks))
```

Python except you have to manage Context explicitly



```
async def make_a_pizza()  
tasks = [...]  
loop = asyncio.get_event_loop()
```

```
loop.run_until_complete(asyncio.wait(tasks))
```

Python except you have to manage Context explicitly
... and define executor for Thread or Process Pool

Review

- Parallelism with javascript async/await
- Efficient Parallelism with Swift and Python

謝謝

Thank You

- Total time on keynote: 150mins
- Presentation: 20mins
- Seminar: 25mins
- ToP = 7.5
- ToS = 6.0
- SoP = 1.25