ASYNC AWAIT #2/2

async & promise

```
const timeout =(f, ms)=>new Promise(res=>setTimeout(_=>res(f()), ms));
const f1 =_=>"abc";
const f2 =_=>"def";
const start = performance.now();
```

```
const timeout =(f, ms)=>new Promise(res=>setTimeout(_=>res(f()), ms));
const f1 =_=>"abc";
const f2 =_=>"def";
const start = performance.now();
```

```
const timeout =(f, ms)=>new Promise(res=>setTimeout(_=>res(f()), ms));
const f1 =_=>"abc";
const f2 =_=>"def";
const start = performance.now();
```

```
const timeout =(f, ms)=>new Promise(res=>setTimeout(_=>res(f()), ms));
const f1 =_=>"abc";
const f2 =_=>"def";
const start = performance.now();

const promise1 = new Promise(res=>res(f1()));
promise1.then(console.log);
```

```
const timeout =(f, ms)=>new Promise(res=>setTimeout(_=>res(f()), ms));
const f1 =_=>"abc";
const f2 =_=>"def";
const start = performance.now();

const promise1 = new Promise(res=>res(f1()));
promise1.then(console.log);

const promise2 =          1
promise2.then(console.log);
```

```
const timeout =(f, ms)=>new Promise(res=>setTimeout(_=>res(f()), ms));
const f1 =_=>"abc";
const f2 =_=>"def";
const start = performance.now();

const promise1 = new Promise(res=>res(f1()));
promise1.then(console.log);

 const promise2 = (async()=>f2())();
 promise2.then(console.log);
```

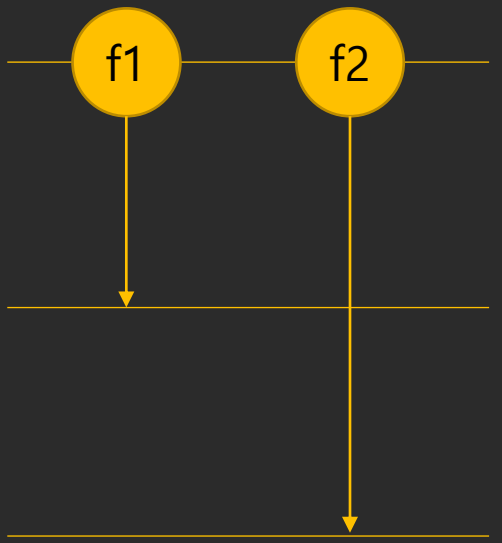
async / await

```
const timeout =(f, ms)=>new Promise(res=>setTimeout(_=>res(f()), ms));
const f1 =_=>"abc";
const f2 =_=>"def";
const start = performance.now();
```

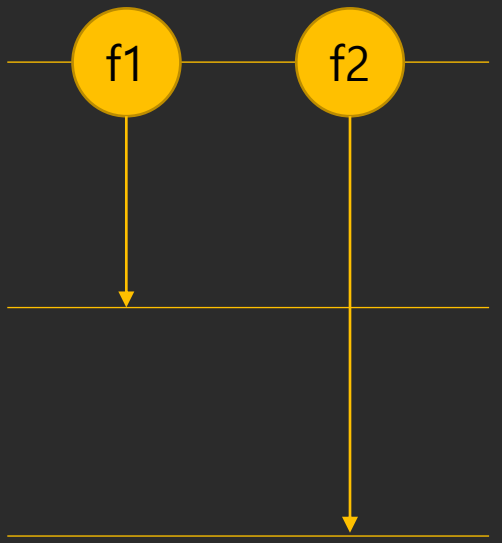
```
const timeout =(f, ms)=>new Promise(res=>setTimeout(_=>res(f()), ms));
const f1 =_=>"abc";
const f2 =_=>"def";
const start = performance.now();

(()=>{
  timeout(f1, 500).then(v=>console.log(v, performance.now() - start));
})();
(()=>{
  timeout(f2, 1000).then(v=>console.log(v, performance.now() - start));
})();
```

```
const timeout =(f, ms)=>new Promise(res=>setTimeout(_=>res(f()), ms));
const f1 =_=>"abc";
const f2 =_=>"def";
const start = performance.now();

(()=>{
  timeout(f1, 500).then(v=>console.log(v, performance.now() - start));
})();
(()=>{
  timeout(f2, 1000).then(v=>console.log(v, performance.now() - start));
})();
```

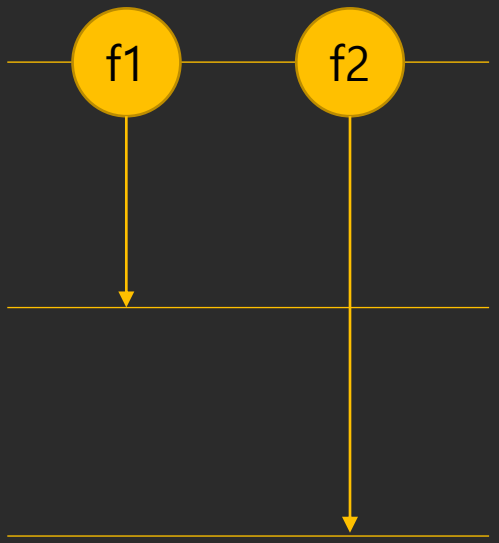
```
const timeout =(f, ms)=>new Promise(res=>setTimeout(_=>res(f()), ms));
const f1 =_=>"abc";
const f2 =_=>"def";
const start = performance.now();

(()=>{
  timeout(f1, 500).then(v=>console.log(v, performance.now() - start));
})();
(()=>{
  timeout(f2, 1000).then(v=>console.log(v, performance.now() - start));
})();
```



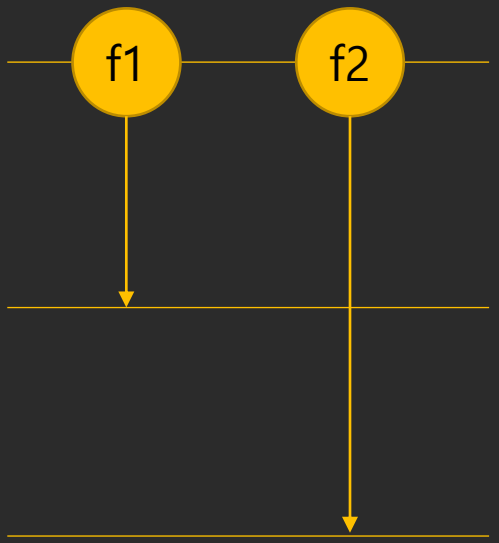
```
const timeout =(f, ms)=>new Promise(res=>setTimeout(_=>res(f()), ms));
const f1 =_=>"abc";
const f2 =_=>"def";
const start = performance.now();

(()=>{
  timeout(f1, 500).then(v=>console.log(v, performance.now() - start));
})();
(()=>{
  timeout(f2, 1000).then(v=>console.log(v, performance.now() - start));
})();

(async ()=>{
  3
})();
```



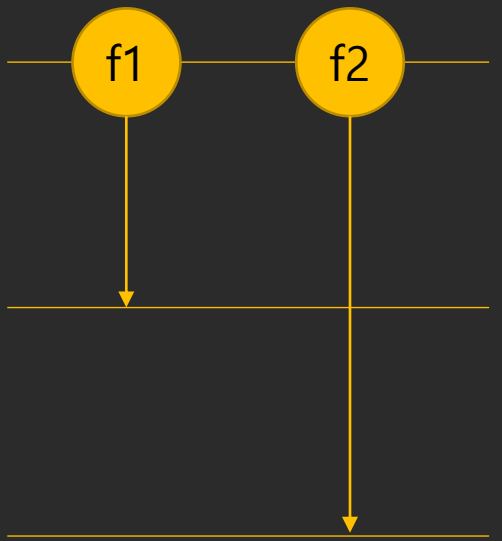
```
const timeout =(f, ms)=>new Promise(res=>setTimeout(_=>res(f()), ms));
const f1 =_=>"abc";
const f2 =_=>"def";
const start = performance.now();

(()=>{
  timeout(f1, 500).then(v=>console.log(v, performance.now() - start));
})();
(()=>{
  timeout(f2, 1000).then(v=>console.log(v, performance.now() - start));
})();

(async ()=>{
  console.log(await timeout(f1, 500), performance.now() - start);
})();
```



```
const timeout =(f, ms)=>new Promise(res=>setTimeout(_=>res(f()), ms));
const f1 =_=>"abc";
const f2 =_=>"def";
const start = performance.now();

(()=>{
  timeout(f1, 500).then(v=>console.log(v, performance.now() - start));
})();
(()=>{
  timeout(f2, 1000).then(v=>console.log(v, performance.now() - start));
})();

(async ()=>{
  console.log(await timeout(f1, 500), performance.now() - start);
})();
(async ()=>{
  console.log(await timeout(f2, 1000), performance.now() - start);
})();
```

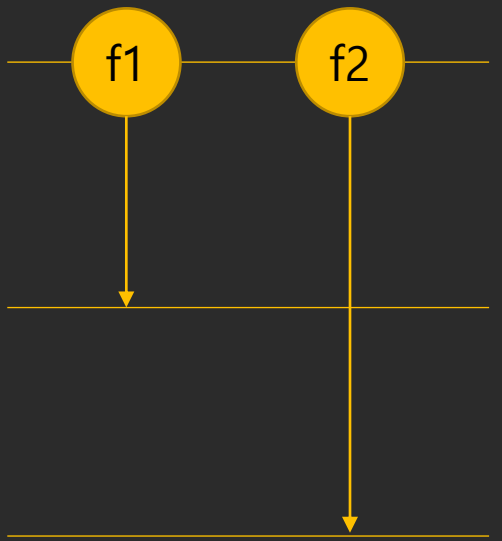


```
const timeout =(f, ms)=>new Promise(res=>setTimeout(_=>res(f()), ms));
const f1 =_=>"abc";
const f2 =_=>"def";
const start = performance.now();
(()=>{
  timeout(f1, 500).then(v=>console.log(v, performance.now() - start));
})();
(()=>{
  timeout(f2, 1000).then(v=>console.log(v, performance.now() - start));
})();
(async ()=>{
  console.log(await timeout(f1, 500), performance.now() - start);
})();
(async ()=>{
  console.log(await timeout(f2, 1000), performance.now() - start);
})();
```



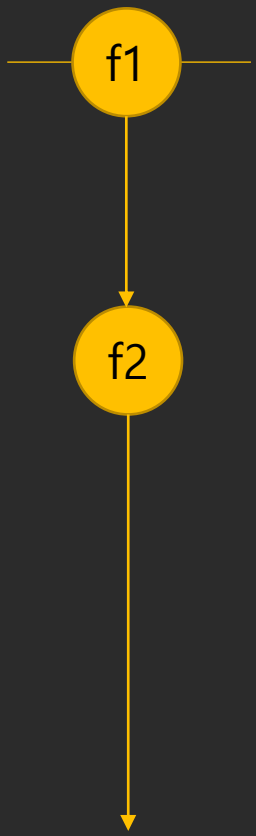
```
const timeout =(f, ms)=>new Promise(res=>setTimeout(_=>res(f()), ms));
const f1 =_=>"abc";
const f2 =_=>"def";
const start = performance.now();

(()=>{
  timeout(f1, 500).then(v=>console.log(v, performance.now() - start));
})();
(()=>{
  timeout(f2, 1000).then(v=>console.log(v, performance.now() - start));
})();

(async ()=>{
  console.log(await timeout(f1, 500), performance.now() - start);
})();
(async ()=>{
  console.log(await timeout(f2, 1000), performance.now() - start);
})();
```

SUSPEND

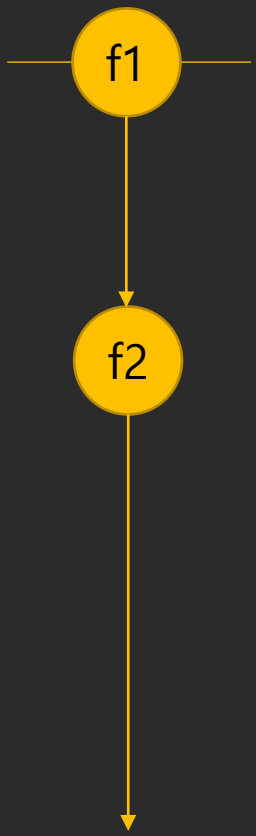


```
const timeout =(f, ms)=>new Promise(res=>setTimeout(_=>res(f()), ms));
const f1 =_=>"abc";
const f2 =_=>"def";
const start = performance.now();

(()=>{
  timeout(f1, 500)
    .then(v=>{
      console.log(v, performance.now() - start);
      return timeout(f2, 1000);
    })
    .then(v=>console.log(v, performance.now() - start));
})();
```

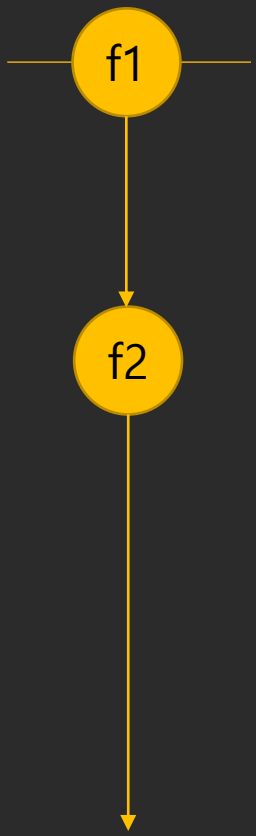
```
const timeout =(f, ms)=>new Promise(res=>setTimeout(_=>res(f()), ms));
const f1 =_=>"abc";
const f2 =_=>"def";
const start = performance.now();

(()=>{
  timeout(f1, 500)
    .then(v=>{
      console.log(v, performance.now() - start);
      return timeout(f2, 1000);
    })
    .then(v=>console.log(v, performance.now() - start));
})();
```



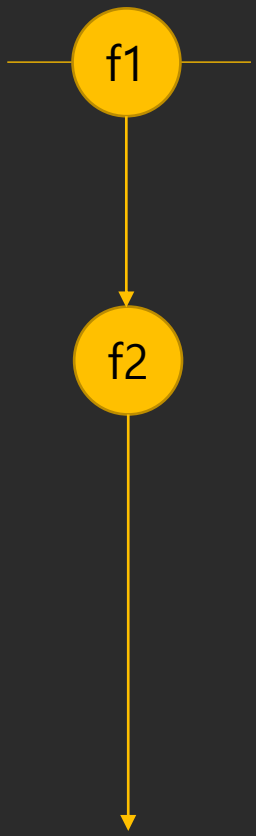
```
const timeout =(f, ms)=>new Promise(res=>setTimeout(_=>res(f()), ms));
const f1 =_=>"abc";
const f2 =_=>"def";
const start = performance.now();

(()=>{
  timeout(f1, 500)
    .then(v=>{
      console.log(v, performance.now() - start);
      return timeout(f2, 1000);
    })
    .then(v=>console.log(v, performance.now() - start));
})();

(async ()=>{
  console.log(await timeout(f1, 500), performance.now() - start);
  console.log(await timeout(f2, 1000), performance.now() - start);
})();
```



```
const timeout =(f, ms)=>new Promise(res=>setTimeout(_=>res(f()), ms));
const f1 =_=>"abc";
const f2 =_=>"def";
const start = performance.now();

(()=>{
  timeout(f1, 500)
    .then(v=>{
      console.log(v, performance.now() - start);
      return timeout(f2, 1000);
    })
    .then(v=>console.log(v, performance.now() - start));
})();

(async ()=>{
  console.log(await timeout(f1, 500), performance.now() - start);
  console.log(await timeout(f2, 1000), performance.now() - start);
})();
```

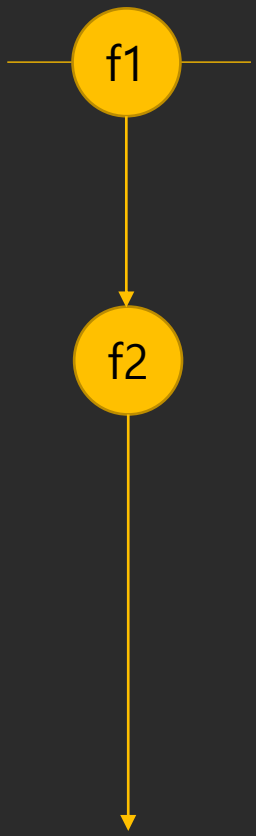


```
const timeout =(f, ms)=>new Promise(res=>setTimeout(_=>res(f()), ms));
const f1 =_=>"abc";
const f2 =_=>"def";
const start = performance.now();

(()=>{
  timeout(f1, 500)
    .then(v=>{
      console.log(v, performance.now() - start);
      return timeout(f2, 1000);
    })
    .then(v=>console.log(v, performance.now() - start));
})();

(async ()=>{
  console.log(await timeout(f1, 500), performance.now() - start);
  console.log(await timeout(f2, 1000), performance.now() - start);
})();
                    SUSPEND
```



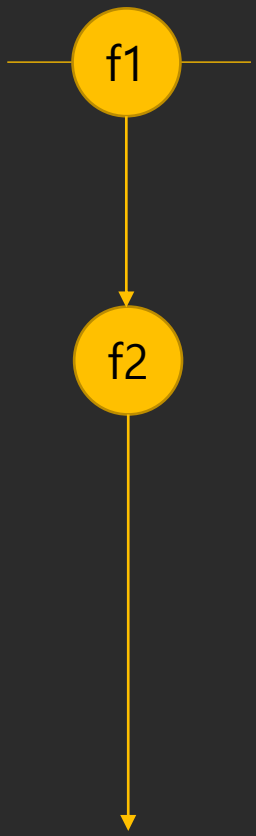
```
const timeout =(f, ms)=>new Promise(res=>setTimeout(_=>res(f()), ms));
const f1 =_=>"abc";
const f2 =_=>"def";
const start = performance.now();

(()=>{
  timeout(f1, 500)
    .then(v=>{
      console.log(v, performance.now() - start);
      return timeout(f2, 1000);
    })
    .then(v=>console.log(v, performance.now() - start));
})();

(async ()=>{
  console.log(await timeout(f1, 500), performance.now() - start);
  console.log(await timeout(f2, 1000), performance.now() - start);
})();
```

SUSPEND



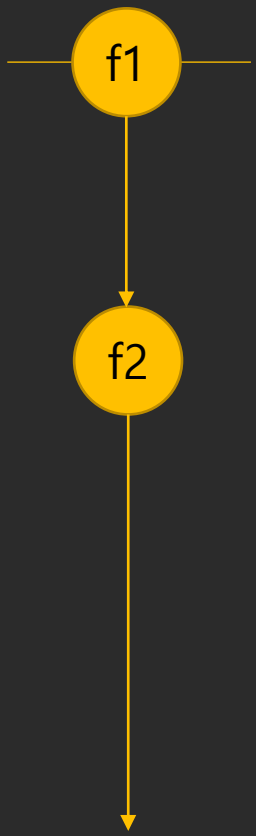
```
const timeout =(f, ms)=>new Promise(res=>setTimeout(_=>res(f()), ms));
const f1 =_=>"abc";
const f2 =_=>"def";
const start = performance.now();

((()=>{
  timeout(f1, 500)
    .then(v=>{
      console.log(v, performance.now() - start);
      return timeout(f2, 1000);
    })
    .then(v=>console.log(v, performance.now() - start));
})();

(async ()=>{
  console.log(await timeout(f1, 500), performance.now() - start);
  console.log(await timeout(f2, 1000), performance.now() - start);
})();
```

SUSPEND

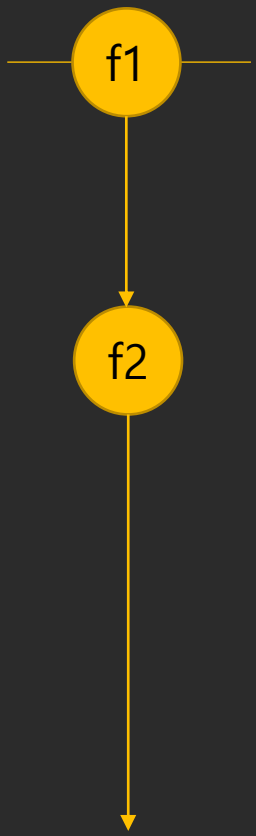


```
const timeout =(f, ms)=>new Promise(res=>setTimeout(_=>res(f()), ms));
const f1 =_=>"abc";
const f2 =_=>"def";
const start = performance.now();

(()=>{
  timeout(f1, 500)
    .then(v=>{
      console.log(v, performance.now() - start);
      return timeout(f2, 1000);
    })
    .then(v=>console.log(v, performance.now() - start));
})();

(async ()=>{
  console.log(await timeout(f1, 500), performance.now() - start);
  console.log(await timeout(f2, 1000), performance.now() - start);
})();
```

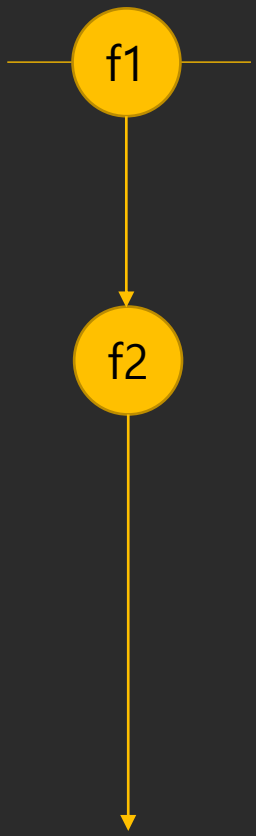
SUSPEND



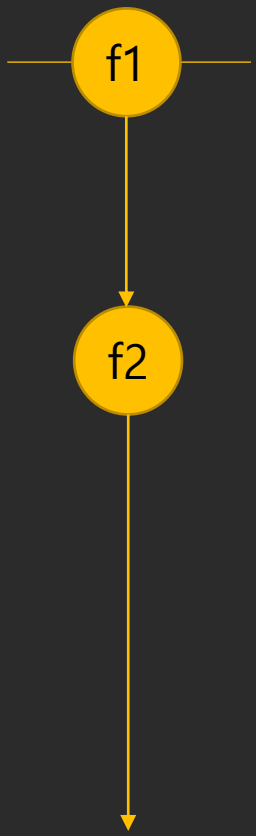
```
const timeout =(f, ms)=>new Promise(res=>setTimeout(_=>res(f()), ms));
const f1 =_=>"abc";
const f2 =_=>"def";
const start = performance.now();

(()=>{
  timeout(f1, 500)
    .then(v=>{
      console.log(v, performance.now() - start);
      return timeout(f2, 1000);
    })
    .then(v=>console.log(v, performance.now() - start));
})();

(async ()=>{
  console.log(
    5
  );
})();
```



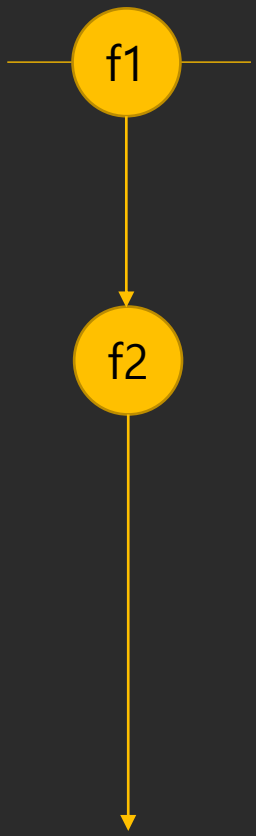
```
const timeout =(f, ms)=>new Promise(res=>setTimeout(_=>res(f()), ms));
const f1 =_=>"abc";
const f2 =_=>"def";
const start = performance.now();

(()=>{
  timeout(f1, 500)
    .then(v=>{
      console.log(v, performance.now() - start);
      return timeout(f2, 1000);
    })
    .then(v=>console.log(v, performance.now() - start));
})();

(async ()=>{
  console.log(
    await timeout(f1, 500), performance.now() - start,
    await timeout(f2, 1000), performance.now() - start
  );
})();
```



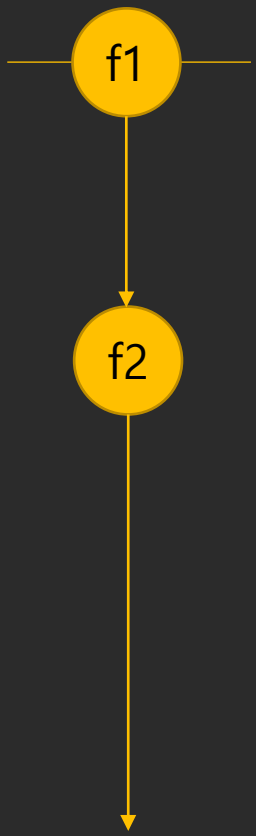
```
const timeout =(f, ms)=>new Promise(res=>setTimeout(_=>res(f()), ms));
const f1 =_=>"abc";
const f2 =_=>"def";
const start = performance.now();

(()=>{
  timeout(f1, 500)
    .then(v=>{
      console.log(v, performance.now() - start);
      return timeout(f2, 1000);
    })
    .then(v=>console.log(v, performance.now() - start));
})();

(async ()=>{
  console.log(
    await timeout(f1, 500), performance.now() - start,
    await timeout(f2, 1000), performance.now() - start
  );
})();
```



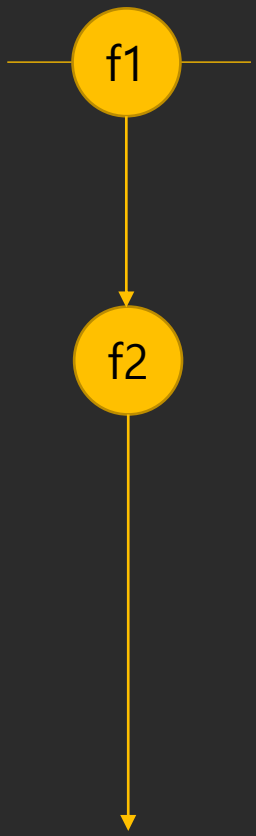
```
const timeout =(f, ms)=>new Promise(res=>setTimeout(_=>res(f()), ms));
const f1 =_=>"abc";
const f2 =_=>"def";
const start = performance.now();

(()=>{
  timeout(f1, 500)
    .then(v=>{
      console.log(v, performance.now() - start);
      return timeout(f2, 1000);
    })
    .then(v=>console.log(v, performance.now() - start));
})();

(async ()=>{
  console.log(
    await timeout(f1, 500), performance.now() - start,
    await timeout(f2, 1000), performance.now() - start
  );        SUSPEND
})();
```



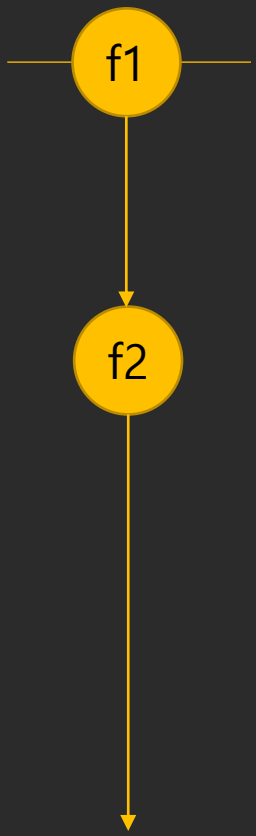
```
const timeout =(f, ms)=>new Promise(res=>setTimeout(_=>res(f()), ms));
const f1 =_=>"abc";
const f2 =_=>"def";
const start = performance.now();

(()=>{
  timeout(f1, 500)
    .then(v=>{
      console.log(v, performance.now() - start);
      return timeout(f2, 1000);
    })
    .then(v=>console.log(v, performance.now() - start));
})();

(async ()=>{
  console.log(
    await timeout(f1, 500), performance.now() - start,
    await timeout(f2, 1000), performance.now() - start
  );        SUSPEND
})();
```



```
const timeout =(f, ms)=>new Promise(res=>setTimeout(_=>res(f()), ms));
const f1 =_=>"abc";
const f2 =_=>"def";
const start = performance.now();

(()=>{
  timeout(f1, 500)
    .then(v=>{
      console.log(v, performance.now() - start);
      return timeout(f2, 1000);
    })
    .then(v=>console.log(v, performance.now() - start));
})();

(async ()=>{
  console.log(
    await timeout(f1, 500), performance.now() - start,
    await timeout(f2, 1000), performance.now() - start
  );        SUSPEND
})();
```



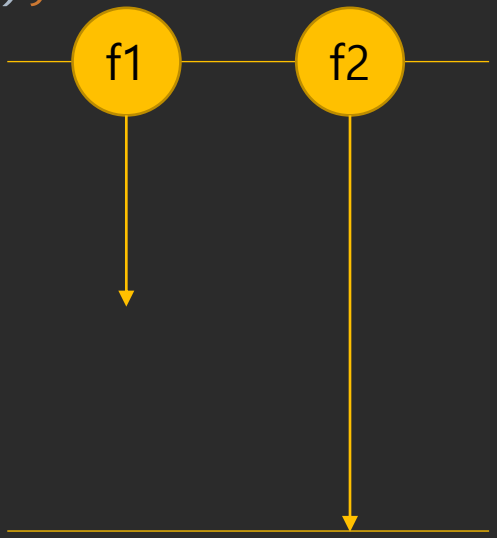
```
const timeout =(f, ms)=>new Promise(res=>setTimeout(_=>res(f()), ms));
const f1 =_=>"abc";
const f2 =_=>"def";
const start = performance.now();

(()=>{
  timeout(f1, 500)
    .then(v=>{
      console.log(v, performance.now() - start);
      return timeout(f2, 1000);
    })
    .then(v=>console.log(v, performance.now() - start));
})();

(async ()=>{
  console.log(
    await timeout(f1, 500), performance.now() - start,
    await timeout(f2, 1000), performance.now() - start
  );        SUSPEND
})();
```



실무에서 Promise.all 과 race를 사용하나요?

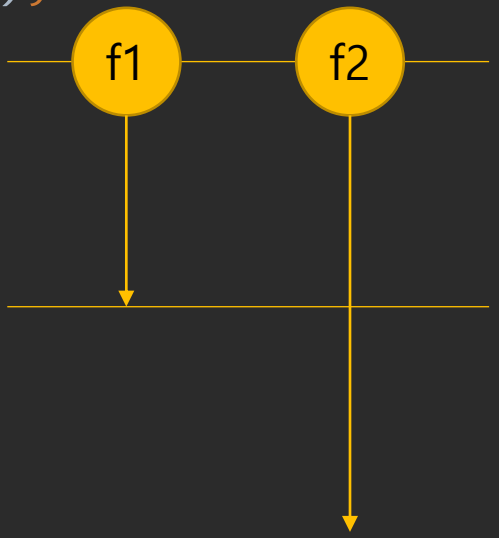
```
const timeout =(f, ms)=>new Promise(res=>setTimeout(_=>res(f()), ms));
const f1 =_=>"abc";
const f2 =_=>"def";
const start = performance.now();

(async ()=>{
  const [v1, v2] = await Promise.all([timeout(f1, 500), timeout(f2, 1000)]);
  console.log(v1, v2, performance.now() - start);
})();
```

```
const timeout =(f, ms)=>new Promise(res=>setTimeout(_=>res(f()), ms));
const f1 =_=>"abc";
const f2 =_=>"def";
const start = performance.now();

(async ()=>{
  const [v1, v2] = await Promise.all([timeout(f1, 500), timeout(f2, 1000)]);
  console.log(v1, v2, performance.now() - start);
})();
```



```
const timeout =(f, ms)=>new Promise(res=>setTimeout(_=>res(f()), ms));
const f1 =_=>"abc";
const f2 =_=>"def";
const start = performance.now();

(async ()=>{
  const [v1, v2] = await Promise.all([timeout(f1, 500), timeout(f2, 1000)]);
  console.log(v1, v2, performance.now() - start);
})();

(async ()=>{
  const v = await Promise.race([timeout(f1, 500), timeout(f2, 1000)]);
  console.log(v, performance.now() - start);
})();
```

```
const timeout =(f, ms)=>new Promise(res=>setTimeout(_=>res(f()), ms));
const f1 =_=>"abc";
const f2 =_=>"def";
const start = performance.now();

(async ()=>{
  const [v1, v2] = await Promise.all([timeout(f1, 500), timeout(f2, 1000)]);
  console.log(v1, v2, performance.now() - start);
})();

(async ()=>{
  const v = await Promise.race([timeout(f1, 500), timeout(f2, 1000)]);
  console.log(v, performance.now() - start);
})();
```



timeout fetch

실무에서 사용 중인 통신방법

```
const api = async(url, timeout = 5000, info = {})=>{
  try {
    let id = -1;
    const v = await Promise.race([
      new Promise(res=>id = window.setTimeout(_=>res(), timeout)),
      fetch(new Request(url, info))
    ]);
    if(v instanceof Response){
      clearTimeout(id);
      return v.status === 404 ? new Error("404") : await v.text();
    }else return new Error("timeout");
  }catch(e){
    return e;
  }
};
```

```
const api = async(url, timeout = 5000, info = {})=>{
  try {
    let id = -1;
    const v = await Promise.race([
      new Promise(res=>id = window.setTimeout(_=>res(), timeout)),
      fetch(new Request(url, info))
    ]);
    if(v instanceof Response){
      clearTimeout(id);
      return v.status === 404 ? new Error("404") : await v.text();
    }else return new Error("timeout");
  }catch(e){
    return e;
  }
};
```

```
const api = async(url, timeout = 5000, info = {})=>{
  try {
    let id = -1;
    const v = await Promise.race([
      new Promise(res=>id = window.setTimeout(_=>res(), timeout)),
      fetch(new Request(url, info))
    ]);
    if(v instanceof Response){
      clearTimeout(id);
      return v.status === 404 ? new Error("404") : await v.text();
    }else return new Error("timeout");
  }catch(e){
    return e;
  }
};
```

```
const api = async(url, timeout = 5000, info = {})=>{
  try {
    let id = -1;
    const v = await Promise.race([
      new Promise(res=>id = window.setTimeout(_=>res(), timeout)),
      fetch(new Request(url, info))
    ]);
    if(v instanceof Response){
      clearTimeout(id);
      return v.status === 404 ? new Error("404") : await v.text();
    }else return new Error("timeout");
  }catch(e){
    return e;
  }
};
```

```
const api = async(url, timeout = 5000, info = {})=>{
  try {
    let id = -1;
    const v = await Promise.race([
      new Promise(res=>id = window.setTimeout(_=>res(), timeout)),
      fetch(new Request(url, info))
    ]);
    if(v instanceof Response){
      clearTimeout(id);
      return v.status === 404 ? new Error("404") : await v.text();
    }else return new Error("timeout");
  }catch(e){
    return e;
  }
};
```

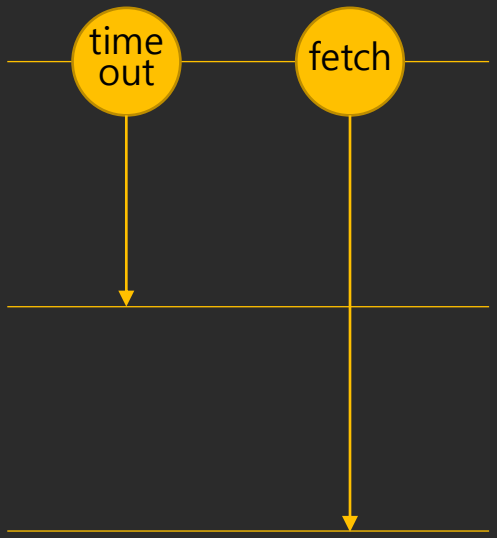
```
const api = async(url, timeout = 5000, info = {})=>{
  try {
    let id = -1;
    const v = await Promise.race([
      new Promise(res=>id = window.setTimeout(_=>res(), timeout)),
      fetch(new Request(url, info))
    ]);
    if(v instanceof Response){
      clearTimeout(id);
      return v.status === 404 ? new Error("404") : await v.text();
    }else return new Error("timeout");
  }catch(e){
    return e;
  }
};
```

```
const api = async(url, timeout = 5000, info = {})=>{
  try {
    let id = -1;
    const v = await Promise.race([
      new Promise(res=>id = window.setTimeout(_=>res(), timeout)),
      fetch(new Request(url, info))
    ]);
    if(v instanceof Response){
      clearTimeout(id);
      return v.status === 404 ? new Error("404") : await v.text();
    }else return new Error("timeout");
  }catch(e){
    return e;
  }
};
```

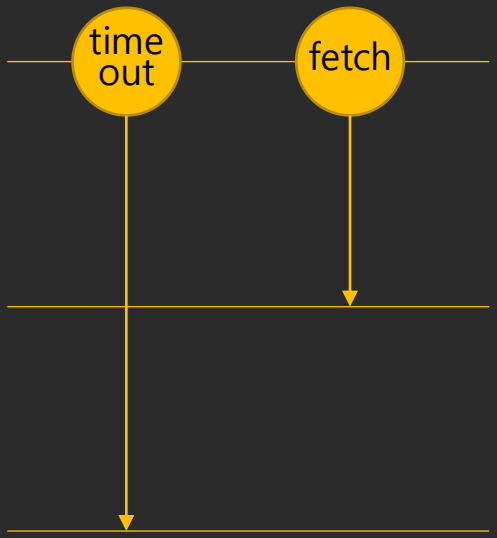
```
const api = async(url, timeout = 5000, info = {})=>{
  try {
    let id = -1;
    const v = await Promise.race([
      new Promise(res=>id = window.setTimeout(_=>res(), timeout)),
      fetch(new Request(url, info))
    ]);
    if(v instanceof Response){
      clearTimeout(id);
      return v.status === 404 ? new Error("404") : await v.text();
    }else return new Error("timeout");
  }catch(e){
    return e;
  }
};
```

```
const api = async(url, timeout = 5000, info = {})=>{
  try {
    let id = -1;
    const v = await Promise.race([
      new Promise(res=>id = window.setTimeout(_=>res(), timeout)),
      fetch(new Request(url, info))
    ]);
    if(v instanceof Response){
      clearTimeout(id);
      return v.status === 404 ? new Error("404") : await v.text();
    }else return new Error("timeout");
  }catch(e){
    return e;
  }
};
```

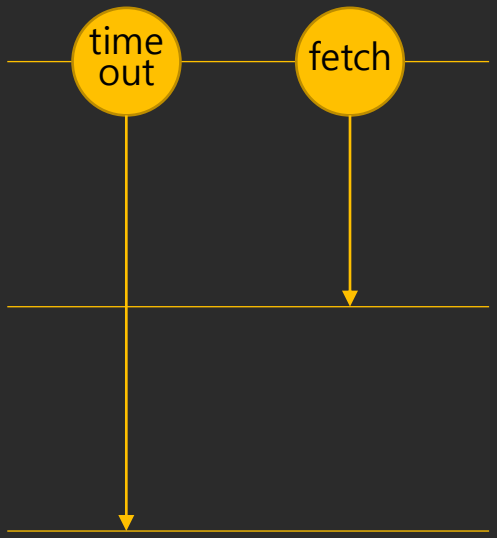
```
const api = async(url, timeout = 5000, info = {})=>{
  try {
    let id = -1;
    const v = await Promise.race([
      new Promise(res=>id = window.setTimeout(_=>res(), timeout)),
      fetch(new Request(url, info))
    ]);
    if(v instanceof Response){
      clearTimeout(id);
      return v.status === 404 ? new Error("404") : await v.text();
    }else return new Error("timeout");
  }catch(e){
    return e;
  }
};
```



```
const api = async(url, timeout = 5000, info = {})=>{
  try {
    let id = -1;
    const v = await Promise.race([
      new Promise(res=>id = window.setTimeout(_=>res(), timeout)),
      fetch(new Request(url, info))
    ]);
    if(v instanceof Response){
      clearTimeout(id);
      return v.status === 404 ? new Error("404") : await v.text();
    }else return new Error("timeout");
  }catch(e){
    return e;
  }
};
```



```
const api = async(url, timeout = 5000, info = {})=>{
  try {
    let id = -1;
    const v = await Promise.race([
      new Promise(res=>id = window.setTimeout(_=>res(), timeout)),
      fetch(new Request(url, info))
    ]);
    if(v instanceof Response){
      clearTimeout(id);
      return v.status === 404 ? new Error("404") : await v.text();
    }else return new Error("timeout");
  }catch(e){
    return e;
  }
};

(async()=>{
  const v = await api("200.html", 1);
  if(v instanceof Error) console.log(`error : ${v}`);
  else console.log(`contents : ${v}`);
})();
```



```
const api2 = async(url, timeout = 5000, info = {})=>{
  let id = -1;
  const v = await Promise.race([
    new Promise(res=>id = window.setTimeout(_=>res(), timeout)),
    fetch(new Request(url, info))
  ]);
  if(v instanceof Response){
    clearTimeout(id);
    if(v.status === 404) throw new Error("404");
    return await v.json();
  }else throw new Error("timeout");
};
```

```
const api2 = async(url, timeout = 5000, info = {})=>{
  let id = -1;
  const v = await Promise.race([
    new Promise(res=>id = window.setTimeout(_=>res(), timeout)),
    fetch(new Request(url, info))
  ]);
  if(v instanceof Response){
    clearTimeout(id);
    if(v.status === 404) throw new Error("404");
    return await v.json();
  }else throw new Error("timeout");
};
```

```
const api2 = async(url, timeout = 5000, info = {})=>{
  let id = -1;
  const v = await Promise.race([
    new Promise(res=>id = window.setTimeout(_=>res(), timeout)),
    fetch(new Request(url, info))
  ]);
  if(v instanceof Response){
    clearTimeout(id);
    if(v.status === 404) throw new Error("404");
    return await v.json();
  }else throw new Error("timeout");
};

(async()=>{
  try {
    const {id, nick, thumb} = await api("/member");
    const [{name, email, sex}, friendsId] = await Promise.all([api(`/detail/${id}`), api(`/friends/${id}`)]);
    updateMember(nick, thumb, name, email, sex);
    updateFriends(
      (await Promise.all(friendsId.map(id => api(`/detail/${id}`)))).map((v, idx)=>({id:friendsId[idx], ...v}))
    );
  } catch (e) {
    console.log(e);
  }
})();
```

```
const api2 = async(url, timeout = 5000, info = {})=>{
  let id = -1;
  const v = await Promise.race([
    new Promise(res=>id = window.setTimeout(_=>res(), timeout)),
    fetch(new Request(url, info))
  ]);
  if(v instanceof Response){
    clearTimeout(id);
    if(v.status === 404) throw new Error("404");
    return await v.json();
  }else throw new Error("timeout");
};

(async()=>{
  try {
    const {id, nick, thumb} = await api("/member");
    const [{name, email, sex}, friendsId] = await Promise.all([api(`/detail/${id}`), api(`/friends/${id}`)]);
    updateMember(nick, thumb, name, email, sex);
    updateFriends(
      (await Promise.all(friendsId.map(id => api(`/detail/${id}`)))).map((v, idx)=>({id:friendsId[idx], ...v}))
    );
  } catch (e) {
    console.log(e);
  }
})();
```

```
const api2 = async(url, timeout = 5000, info = {})=>{
  let id = -1;
  const v = await Promise.race([
    new Promise(res=>id = window.setTimeout(_=>res(), timeout)),
    fetch(new Request(url, info))
  ]);
  if(v instanceof Response){
    clearTimeout(id);
    if(v.status === 404) throw new Error("404");
    return await v.json();
  }else throw new Error("timeout");
};

(async()=>{
  try {
    const {id, nick, thumb} = await api("/member");
    const [{name, email, sex}, friendsId] = await Promise.all([api(`/detail/${id}`), api(`/friends/${id}`)]);
    updateMember(nick, thumb, name, email, sex);
    updateFriends(
      (await Promise.all(friendsId.map(id => api(`/detail/${id}`)))).map((v, idx)=>({id:friendsId[idx], ...v}))
    );
  } catch (e) {
    console.log(e);
  }
})();
```

```
const api2 = async(url, timeout = 5000, info = {})=>{
  let id = -1;
  const v = await Promise.race([
    new Promise(res=>id = window.setTimeout(_=>res(), timeout)),
    fetch(new Request(url, info))
  ]);
  if(v instanceof Response){
    clearTimeout(id);
    if(v.status === 404) throw new Error("404");
    return await v.json();
  }else throw new Error("timeout");
};

(async()=>{
  try {
    const {id, nick, thumb} = await api("/member");
    const [{name, email, sex}, friendsId] = await Promise.all([api(`/detail/${id}`), api(`/friends/${id}`)]);
    updateMember(nick, thumb, name, email, sex);
    updateFriends(
      (await Promise.all(friendsId.map(id => api(`/detail/${id}`)))).map((v, idx)=>({id:friendsId[idx], ...v}))
    );
  } catch (e) {
    console.log(e);
  }
})();
```

```
const api2 = async(url, timeout = 5000, info = {})=>{
  let id = -1;
  const v = await Promise.race([
    new Promise(res=>id = window.setTimeout(_=>res(), timeout)),
    fetch(new Request(url, info))
  ]);
  if(v instanceof Response){
    clearTimeout(id);
    if(v.status === 404) throw new Error("404");
    return await v.json();
  }else throw new Error("timeout");
};

(async()=>{
  try {
    const {id, nick, thumb} = await api("/member");
    const [{name, email, sex}, friendsId] = await Promise.all([api(`/detail/${id}`), api(`/friends/${id}`)]);
    updateMember(nick, thumb, name, email, sex);
    updateFriends(
      (await Promise.all(friendsId.map(id => api(`/detail/${id}`)))).map((v, idx)=>({id:friendsId[idx], ...v}))
    );
  } catch (e) {
    console.log(e);
  }
})();
```

```
const api2 = async(url, timeout = 5000, info = {})=>{
  let id = -1;
  const v = await Promise.race([
    new Promise(res=>id = window.setTimeout(_=>res(), timeout)),
    fetch(new Request(url, info))
  ]);
  if(v instanceof Response){
    clearTimeout(id);
    if(v.status === 404) throw new Error("404");
    return await v.json();
  }else throw new Error("timeout");
};

(async()=>{
  try {
    const {id, nick, thumb} = await api("/member");
    const [{name, email, sex}, friendsId] = await Promise.all([api(`/detail/${id}`), api(`/friends/${id}`)]);
    updateMember(nick, thumb, name, email, sex);
    updateFriends(
      (await Promise.all(friendsId.map(id => api(`/detail/${id}`)))).map((v, idx)=>({id:friendsId[idx], ...v}))
    );
  } catch (e) {
    console.log(e);
  }
})();
```

```
const api2 = async(url, timeout = 5000, info = {})=>{
  let id = -1;
  const v = await Promise.race([
    new Promise(res=>id = window.setTimeout(_=>res(), timeout)),
    fetch(new Request(url, info))
  ]);
  if(v instanceof Response){
    clearTimeout(id);
    if(v.status === 404) throw new Error("404");
    return await v.json();
  }else throw new Error("timeout");
};

(async()=>{
  try {
    const {id, nick, thumb} = await api("/member");
    const [{name, email, sex}, friendsId] = await Promise.all([api(`/detail/${id}`), api(`/friends/${id}`)]);
    updateMember(nick, thumb, name, email, sex);
    updateFriends(
      (await Promise.all(friendsId.map(id => api(`/detail/${id}`)))).map((v, idx)=>({id:friendsId[idx], ...v}))
    );
  } catch (e) {
    console.log(e);
  }
})();
```

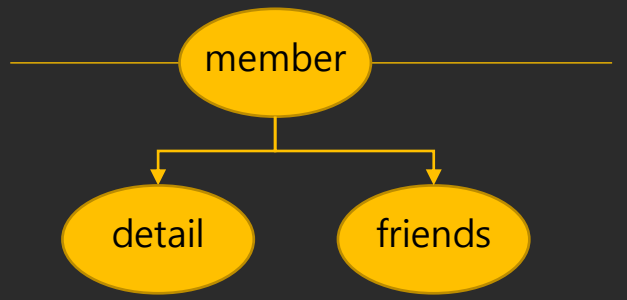
```
const api2 = async(url, timeout = 5000, info = {})=>{
  let id = -1;
  const v = await Promise.race([
    new Promise(res=>id = window.setTimeout(_=>res(), timeout)),
    fetch(new Request(url, info))
  ]);
  if(v instanceof Response){
    clearTimeout(id);
    if(v.status === 404) throw new Error("404");
    return await v.json();
  }else throw new Error("timeout");
};

(async()=>{
  try {
    const {id, nick, thumb} = await api("/member");
    const [{name, email, sex}, friendsId] = await Promise.all([api(`/detail/${id}`), api(`/friends/${id}`)]);
    updateMember(nick, thumb, name, email, sex);
    updateFriends(
      (await Promise.all(friendsId.map(id => api(`/detail/${id}`)))).map((v, idx)=>({id:friendsId[idx], ...v}))
    );
  } catch (e) {
    console.log(e);
  }
})();
```

```
const api2 = async(url, timeout = 5000, info = {})=>{
  let id = -1;
  const v = await Promise.race([
    new Promise(res=>id = window.setTimeout(_=>res(), timeout)),
    fetch(new Request(url, info))
  ]);
  if(v instanceof Response){
    clearTimeout(id);
    if(v.status === 404) throw new Error("404");
    return await v.json();
  }else throw new Error("timeout");
};

(async()=>{
  try {
    const {id, nick, thumb} = await api("/member");
    const [{name, email, sex}, friendsId] = await Promise.all([api(`/detail/${id}`), api(`/friends/${id}`)]);
    updateMember(nick, thumb, name, email, sex);
    updateFriends(
      (await Promise.all(friendsId.map(id => api(`/detail/${id}`)))).map((v, idx)=>({id:friendsId[idx], ...v}))
    );
  } catch (e) {
    console.log(e);
  }
})();
```

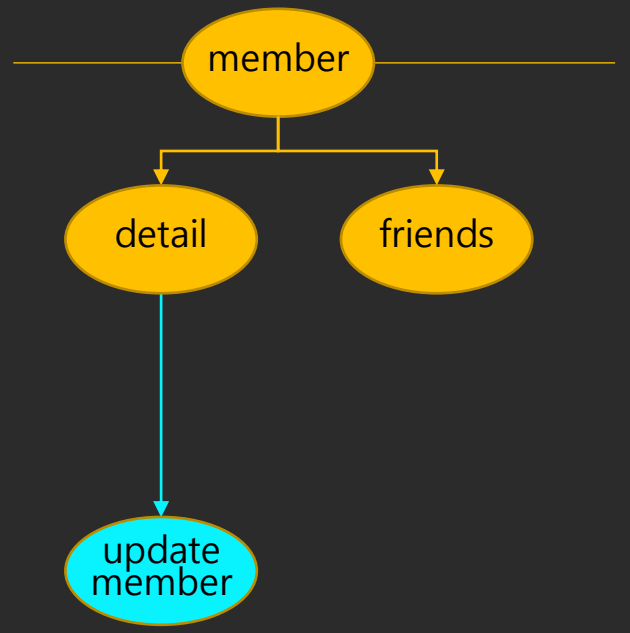
member

```
const api2 = async(url, timeout = 5000, info = {})=>{
  let id = -1;
  const v = await Promise.race([
    new Promise(res=>id = window.setTimeout(_=>res(), timeout)),
    fetch(new Request(url, info))
  ]);
  if(v instanceof Response){
    clearTimeout(id);
    if(v.status === 404) throw new Error("404");
    return await v.json();
  }else throw new Error("timeout");
};

(async()=>{
  try {
    const {id, nick, thumb} = await api("/member");
    const [{name, email, sex}, friendsId] = await Promise.all([api(`/detail/${id}`), api(`/friends/${id}`)]);
    updateMember(nick, thumb, name, email, sex);
    updateFriends(
      (await Promise.all(friendsId.map(id => api(`/detail/${id}`)))).map((v, idx)=>({id:friendsId[idx], ...v}))
    );
  } catch (e) {
    console.log(e);
  }
})();
```

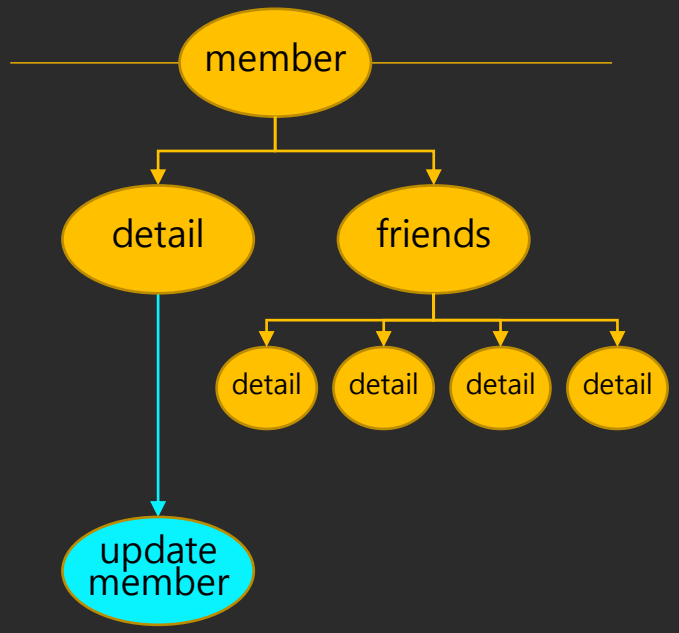
```
const api2 = async(url, timeout = 5000, info = {})=>{
  let id = -1;
  const v = await Promise.race([
    new Promise(res=>id = window.setTimeout(_=>res(), timeout)),
    fetch(new Request(url, info))
  ]);
  if(v instanceof Response){
    clearTimeout(id);
    if(v.status === 404) throw new Error("404");
    return await v.json();
  }else throw new Error("timeout");
};

(async()=>{
  try {
    const {id, nick, thumb} = await api("/member");
    const [{name, email, sex}, friendsId] = await Promise.all([api(`/detail/${id}`), api(`/friends/${id}`)]);
    updateMember(nick, thumb, name, email, sex);
    updateFriends(
      (await Promise.all(friendsId.map(id => api(`/detail/${id}`)))).map((v, idx)=>({id:friendsId[idx], ...v}))
    );
  } catch (e) {
    console.log(e);
  }
})();
```



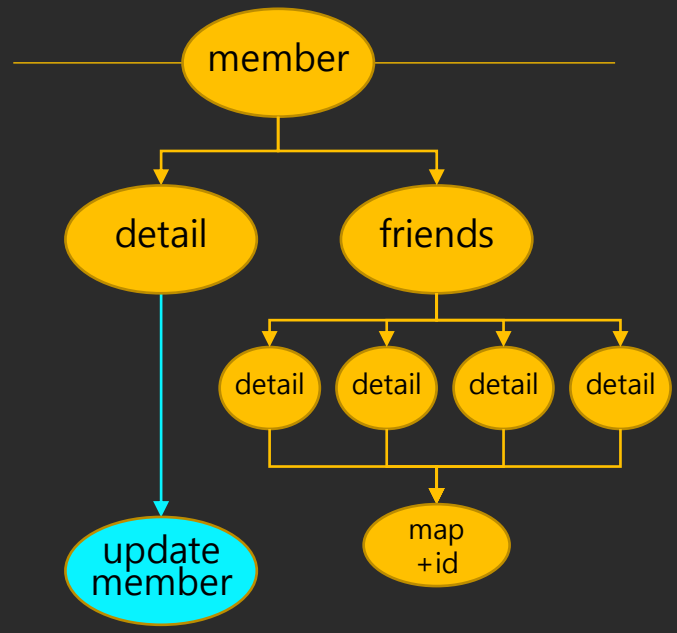
```
const api2 = async(url, timeout = 5000, info = {})=>{
  let id = -1;
  const v = await Promise.race([
    new Promise(res=>id = window.setTimeout(_=>res(), timeout)),
    fetch(new Request(url, info))
  ]);
  if(v instanceof Response){
    clearTimeout(id);
    if(v.status === 404) throw new Error("404");
    return await v.json();
  }else throw new Error("timeout");
};

(async()=>{
  try {
    const {id, nick, thumb} = await api("/member");
    const [{name, email, sex}, friendsId] = await Promise.all([api(`/detail/${id}`), api(`/friends/${id}`)]);
    updateMember(nick, thumb, name, email, sex);
    updateFriends(
      (await Promise.all(friendsId.map(id => api(`/detail/${id}`)))).map((v, idx)=>({id:friendsId[idx], ...v}))
    );
  } catch (e) {
    console.log(e);
  }
})();
```



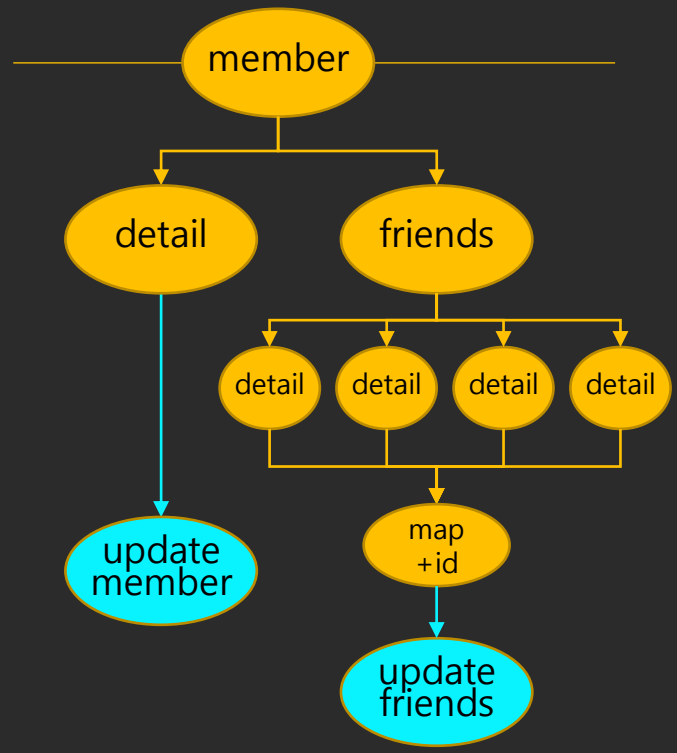
```
const api2 = async(url, timeout = 5000, info = {})=>{
  let id = -1;
  const v = await Promise.race([
    new Promise(res=>id = window.setTimeout(_=>res(), timeout)),
    fetch(new Request(url, info))
  ]);
  if(v instanceof Response){
    clearTimeout(id);
    if(v.status === 404) throw new Error("404");
    return await v.json();
  }else throw new Error("timeout");
};

(async()=>{
  try {
    const {id, nick, thumb} = await api("/member");
    const [{name, email, sex}, friendsId] = await Promise.all([api(`/detail/${id}`), api(`/friends/${id}`)]);
    updateMember(nick, thumb, name, email, sex);
    updateFriends(
      (await Promise.all(friendsId.map(id => api(`/detail/${id}`)))).map((v, idx)=>({id:friendsId[idx], ...v}))
    );
  } catch (e) {
    console.log(e);
  }
})();
```



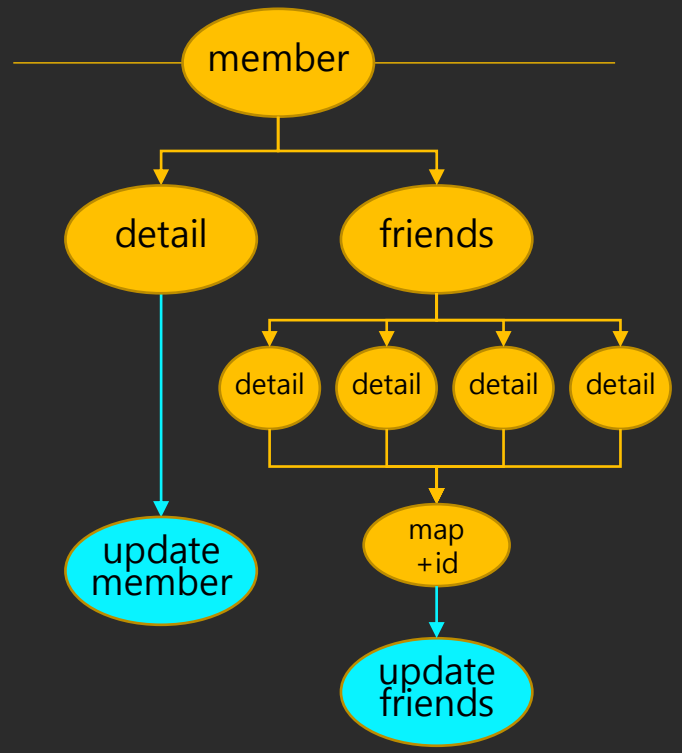
```
const api2 = async(url, timeout = 5000, info = {})=>{
  let id = -1;
  const v = await Promise.race([
    new Promise(res=>id = window.setTimeout(_=>res(), timeout)),
    fetch(new Request(url, info))
  ]);
  if(v instanceof Response){
    clearTimeout(id);
    if(v.status === 404) throw new Error("404");
    return await v.json();
  }else throw new Error("timeout");
};

(async()=>{
  try {
    const {id, nick, thumb} = await api("/member");
    const [{name, email, sex}, friendsId] = await Promise.all([api(`/detail/${id}`), api(`/friends/${id}`)]);
    updateMember(nick, thumb, name, email, sex);
    updateFriends(
      (await Promise.all(friendsId.map(id => api(`/detail/${id}`)))).map((v, idx)=>({id:friendsId[idx], ...v}))
    );
  } catch (e) {
    console.log(e);
  }
})();
```



```
const api2 = async(url, timeout = 5000, info = {})=>{
  let id = -1;
  const v = await Promise.race([
    new Promise(res=>id = window.setTimeout(_=>res(), timeout)),
    fetch(new Request(url, info))
  ]);
  if(v instanceof Response){
    clearTimeout(id);
    if(v.status === 404) throw new Error("404");
    return await v.json();
  }else throw new Error("timeout");
};

(async()=>{
  try {
    const {id, nick, thumb} = await api("/member");
    const [{name, email, sex}, friendsId] = await Promise.all([api(`/detail/${id}`), api(`/friends/${id}`)]);
    updateMember(nick, thumb, name, email, sex);
    updateFriends(
      (await Promise.all(friendsId.map(id => api(`/detail/${id}`)))).map((v, idx)=>({id:friendsId[idx], ...v}))
    );
  } catch (e) {
    console.log(e);
  }
})();
```



```
const api2 = async(url, timeout = 5000, info = {})=>{
  let id = -1;
  const v = await Promise.race([
    new Promise(res=>id = window.setTimeout(_=>res(), timeout)),
    fetch(new Request(url, info))
  ]);
  if(v instanceof Response){
    clearTimeout(id);
    if(v.status === 404) throw new Error("404");
    return await v.text();
  }else throw new Error("timeout");
};

(async()=>{
  try {
    const {id, nick, thumb} = await api("/member");
    const [{name, email, sex}, friendsId] = await Promise.all([api(`/detail/${id}`), api(`/friends/${id}`)]);
    updateMember(nick, thumb, name, email, sex);
    updateFriends(
      (await Promise.all(friendsId.map(id => api(`/detail/${id}`)))).map((v, idx)=>({id:friendsId[idx], ...v}))
    );
  } catch (e) {
    console.log(e);
  }
})();

const updateMember = (nick, thumb, name, email, sex)=>{};
const updateFriends = (details)=>details.map(({id, name, email, sex})=>{});
```



asynchous iterator

실무에서 iterator를 사용하나요?

```
const infinity = async function*(cat){
  let page = -1;
  do {
    try {
      const {nextPage, items} = await api2(`/list/${cat}/${page === -1 ? "" : page}`);
      page = nextPage;
      yield items;
    }catch(e){
      return;
    }
  }while(page !== -1);
};
```

```
const infinity = async function*(cat){
  let page = -1;
  do {
    try {
      const {nextPage, items} = await api2(`/list/${cat}/${page === -1 ? "" : page}`);
      page = nextPage;
      yield items;
    }catch(e){
      return;
    }
  }while(page !== -1);
};
```

```
const infinity = async function*(cat){
  let page = -1;
  do {
    try {
      const {nextPage, items} = await api2(`/list/${cat}/${page === -1 ? "" : page}`);
      page = nextPage;
      yield items;
    }catch(e){
      return;
    }
  }while(page !== -1);
};
```

```
const infinity = async function*(cat){
  let page = -1;
  do {
    try {
      const {nextPage, items} = await api2(`/list/${cat}/${page === -1 ? "" : page}`);
      page = nextPage;
      yield items;
    }catch(e){
      return;
    }
  }while(page !== -1);
};
```

```
const infinity = async function*(cat){
  let page = -1;
  do {
    try {
      const {nextPage, items} = await api2(`/list/${cat}/${page === -1 ? "" : page}`);
      page = nextPage;
      yield items;
    }catch(e){
      return;
    }
  }while(page !== -1);
};
```

```
const infinity = async function*(cat){
  let page = -1;
  do {
    try {
      const {nextPage, items} = await api2(`/list/${cat}/${page === -1 ? "" : page}`);
      page = nextPage;
      yield items;
    }catch(e){
      return;
    }
  }while(page !== -1);
};
```

```
const infinity = async function*(cat){
  let page = -1;
  do {
    try {
      const {nextPage, items} = await api2(`/list/${cat}/${page === -1 ? "" : page}`);
      page = nextPage;
      yield items;
    }catch(e){
      return;
    }
  }while(page !== -1);
};

const notice = infinity("notice");
(async()=>{
  const {value, done} = await notice.next();
  if(!done) console.log(value);
})();
document.querySelector("#a").onclick = async()=>{
  const {value, done} = await notice.next();
  if(!done) console.log(value);
};
```

```
const infinity = async function*(cat){
  let page = -1;
  do {
    try {
      const {nextPage, items} = await api2(`/list/${cat}/${page === -1 ? "" : page}`);
      page = nextPage;
      yield items;
    }catch(e){
      return;
    }
  }while(page !== -1);
};

const notice = infinity("notice");
(async()=>{
  const {value, done} = await notice.next();
  if(!done) console.log(value);
})();
document.querySelector("#a").onclick = async()=>{
  const {value, done} = await notice.next();
  if(!done) console.log(value);
};
```

```
const infinity = async function*(cat){
  let page = -1;
  do {
    try {
      const {nextPage, items} = await api2(`/list/${cat}/${page === -1 ? "" : page}`);
      page = nextPage;
      yield items;
    }catch(e){
      return;
    }
  }while(page !== -1);
};

const notice = infinity("notice");
(async()=>{
  const {value, done} = await notice.next();
  if(!done) console.log(value);
})();
document.querySelector("#next").onclick = async()=>{
  const {value, done} = await notice.next();
  if(!done) console.log(value);
};
```