Offline Lecture 1.5

Async functions vs sync functions, real use of callbacks

JS Browser architecture

Promises

Async await

Async functions vs sync functions

What does synchronous mean?

Together, one after the other, sequential

Only one thing is happening at a time

What does asynchronous mean?

Opposite of synchronous

Happens in parts

Multiple things are context switching with each other

Async functions vs sync functions

Lets build some intuition

Human brain and body is single threaded

1. **We can only do one thing at a time**
2. **But we can context switch b/w tasks, or we can delegate tasks to other people**

Async functions vs sync functions

Lets build some intuition

Human brain and body is single threaded

1. **We can only do one thing at a time**
2. **But we can context switch b/w tasks, or we can delegate tasks to other people**

You have 4 tasks -

1. **Boil water**
2. **Cut vegetables**
3. **Cut maggi packet**
4. **Get ketchup from the shop nearby**

How would you do this? Synchronously or Asynchronously?

Async functions vs sync functions

Lets build some intuition

Lets say you have 4 tasks to make food -

1. **Boil water**
2. **Cut vegetables**
3. **Open maggi packet**
4. **Get ketchup from the shop nearby**

How would you do them?

Async functions vs sync functions

Lets build some intuition

Total time
5 +

5 minutes

Async functions vs sync functions

Lets build some intuition

Total time
5 +

2 minutes

Async functions vs sync functions

Lets build some intuition

Total time
5 + 2

2 minutes

Async functions vs sync functions

Lets build some intuition

Total time
5 + 2 +10

10 minutes

Async functions vs sync functions

Lets build some intuition

Total time
5 + 2 +10 +20
= 37 mins

20 minutes

Async functions vs sync functions

Lets build some intuition

Or, you could multi-task
(More technically, context switch and delegate)

Async functions vs sync functions

Lets build some intuition

Total time

Turn on gas (2 seconds)

Async functions vs sync functions

Lets build some intuition

Total time

Delegate task to house help

Async functions vs sync functions

Lets build some intuition

Total time

Delegate task to house help (kamla didi)

Async functions vs sync functions

Lets build some intuition

Total time

Cut Maggi packet (2 mins)

On the way to get ketchup (2 mins)

Async functions vs sync functions

Lets build some intuition

Total time

Check on water (1s)

On the way to get ketchup (2 mins 1 s)

Async functions vs sync functions

Lets build some intuition

Total time

Cutting vegetables (8 mins)

She’s back with ketchup (10 mins)

Async functions vs sync functions

Lets build some intuition

Total time

Cut vegetables (10 mins)

Waits for 2 mins for u to finish

Async functions vs sync functions

Lets build some intuition

Total time
12 minutes

Everything is done!

Async functions vs sync functions

Lets build some intuition

Turn on gas(1s)
Delegate task

Total time
12 minutes

Cut Maggi packet (2 mins)
Cut vegetables (10 mins)

Get ketchup (10 mins)

Async functions vs sync functions

What did we learn?

Even if you are single threaded (brain can do only one thing at a time), you can do things parallely by Delegating
You can also context switch between tasks if need be (the net time to do both the things would still be the same)

Net amount of time take to do a task can be decreased
by doing these two things (delegating and context switching)

Async functions vs sync functions

How does JS do the same? Can JS delegate? Can JS context switch?

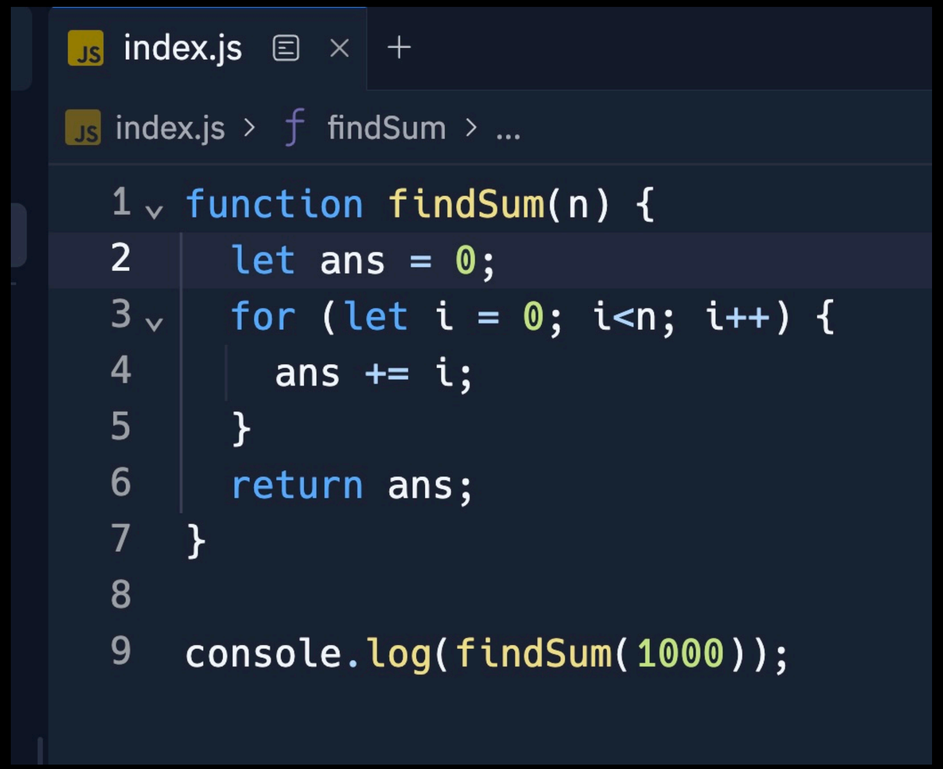
Async functions vs sync functions

How does JS do the same? Can JS delegate? Can JS context switch?

Yes! Using asynchronous functions

Async functions vs sync functions

Until now, we've only seen synchronous functions



Async functions vs sync functions

Lets introduce an asynchronous function (setTimeout)

https://gist.github.com/hkirat/a75987a32fdfcab27672410930327f1a

```
1  function findSum(n) {
2    let ans = 0;
3    for (let i = 0; i<n; i++) {
4      ans += i;
5    }
6    return ans;
7  }
8
9  function findSumTill100() {
10   return findSum(100);
11 }
12
13 setTimeout(findSumTill100, 1000)
14 console.log("hello world");
```

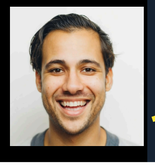
Calling an async function →

Async functions vs sync functions

Lets introduce an asynchronous function (setTimeout)

https://gist.github.com/hkirat/a75987a32fdfcab27672410930327f1a

```
1  function findSum(n) {
2    let ans = 0;
3    for (let i = 0; i<n; i++) {
4      ans += i;
5    }
6    return ans;
7  }
8
9  function findSumTill100() {
10   return findSum(100);
11 }
12
13 setTimeout(findSumTill100, 1000)
14 console.log("hello world");
```



Calling an async function

After 1 second,
remind me to run a function

Async functions vs sync functions

Lets introduce an asynchronous function (setTimeout)

https://gist.github.com/hkirat/a75987a32fdfcab27672410930327f1a

```
1  function findSum(n) {
2    let ans = 0;
3    for (let i = 0; i<n; i++) {
4      ans += i;
5    }
6    return ans;
7  }
8
9  function findSumTill100() {
10   return findSum(100);
11 }
12
13 setTimeout(findSumTill100, 1000)
14 console.log("hello world");
```

Maintains a clock for 1s

Proceeds to to his own thing

Async functions vs sync functions

Lets introduce an asynchronous function (setTimeout)

https://gist.github.com/hkirat/a75987a32fdfcab27672410930327f1a

```
1  function findSum(n) {
2    let ans = 0;
3    for (let i = 0; i<n; i++) {
4      ans += i;
5    }
6    return ans;
7  }
8
9  function findSumTill100() {
10   return findSum(100);
11 }
12
13 setTimeout(findSumTill100, 1000)
14 console.log("hello world");
```

Still running 1s clock

Done with his own thing, sitting idle

Async functions vs sync functions

Lets introduce an asynchronous function (setTimeout)

https://gist.github.com/hkirat/a75987a32fdfcab27672410930327f1a

```
1  function findSum(n) {
2    let ans = 0;
3    for (let i = 0; i<n; i++) {
4      ans += i;
5    }
6    return ans;
7  }
8
9  function findSumTill100() {
10   return findSum(100);
11 }
12
13 setTimeout(findSumTill100, 1000)
14 console.log("hello world");
```

Calls the callback

Async functions vs sync functions

Lets introduce an asynchronous function (setTimeout)

https://gist.github.com/hkirat/a75987a32fdfcab27672410930327f1a

Starts executing the callback

```
1  function findSum(n) {
2    let ans = 0;
3    for (let i = 0; i<n; i++) {
4      ans += i;
5    }
6    return ans;
7  }
8
9  function findSumTill100() {
10   return findSum(100);
11 }
12
13 setTimeout(findSumTill100, 1000)
14 console.log("hello world");
```

Done with their task!

Async functions vs sync functions

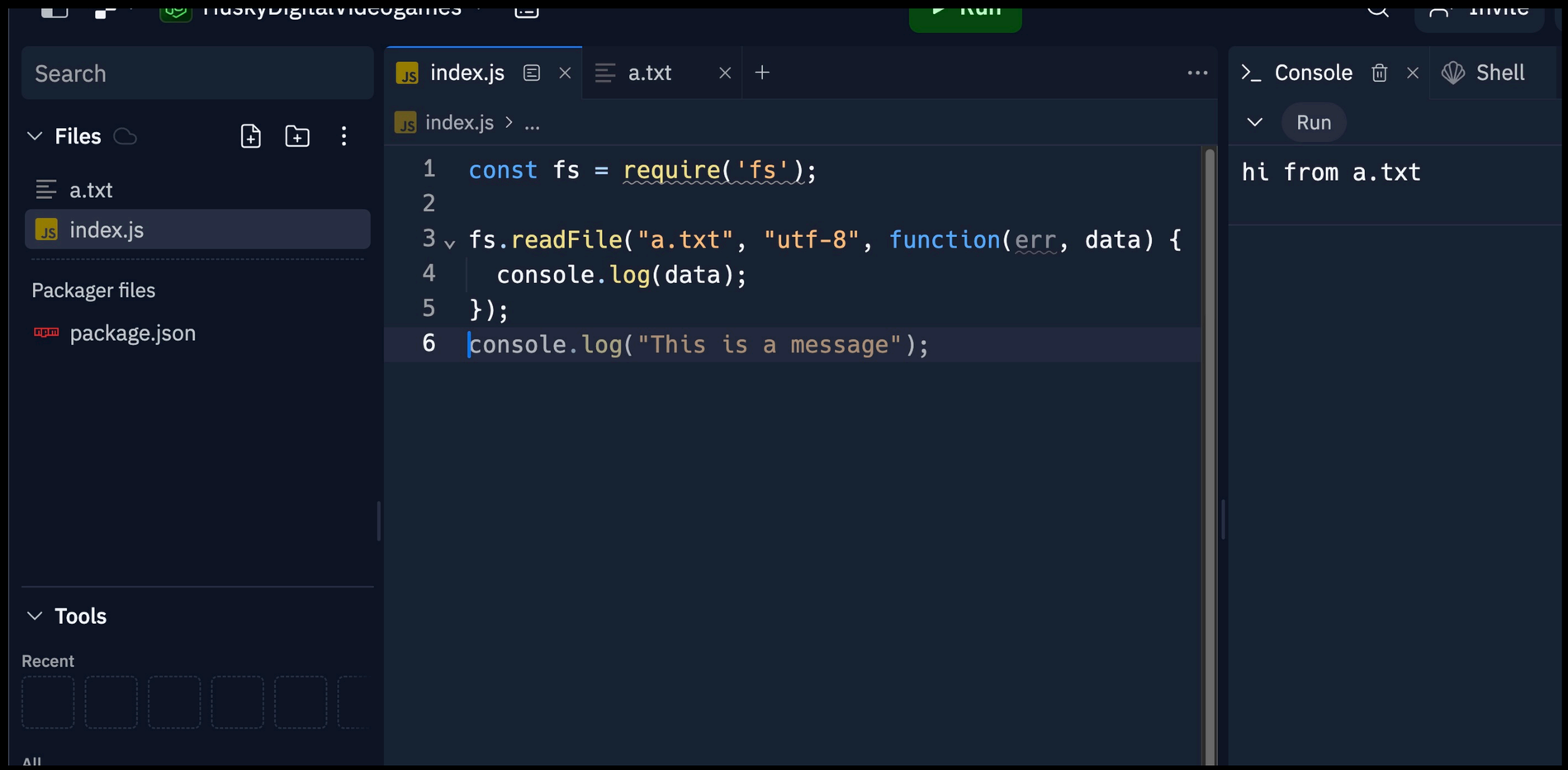
What are common async functions?
setTimeout
fs.readFile - to read a file from your filesystem
Fetch - to fetch some data from an API endpoint

Things that are delegated

Async functions vs sync functions

Lets try fs to read a file

https://gist.github.com/hkirat/8e85f1bc53207ea878801c9f18aab016

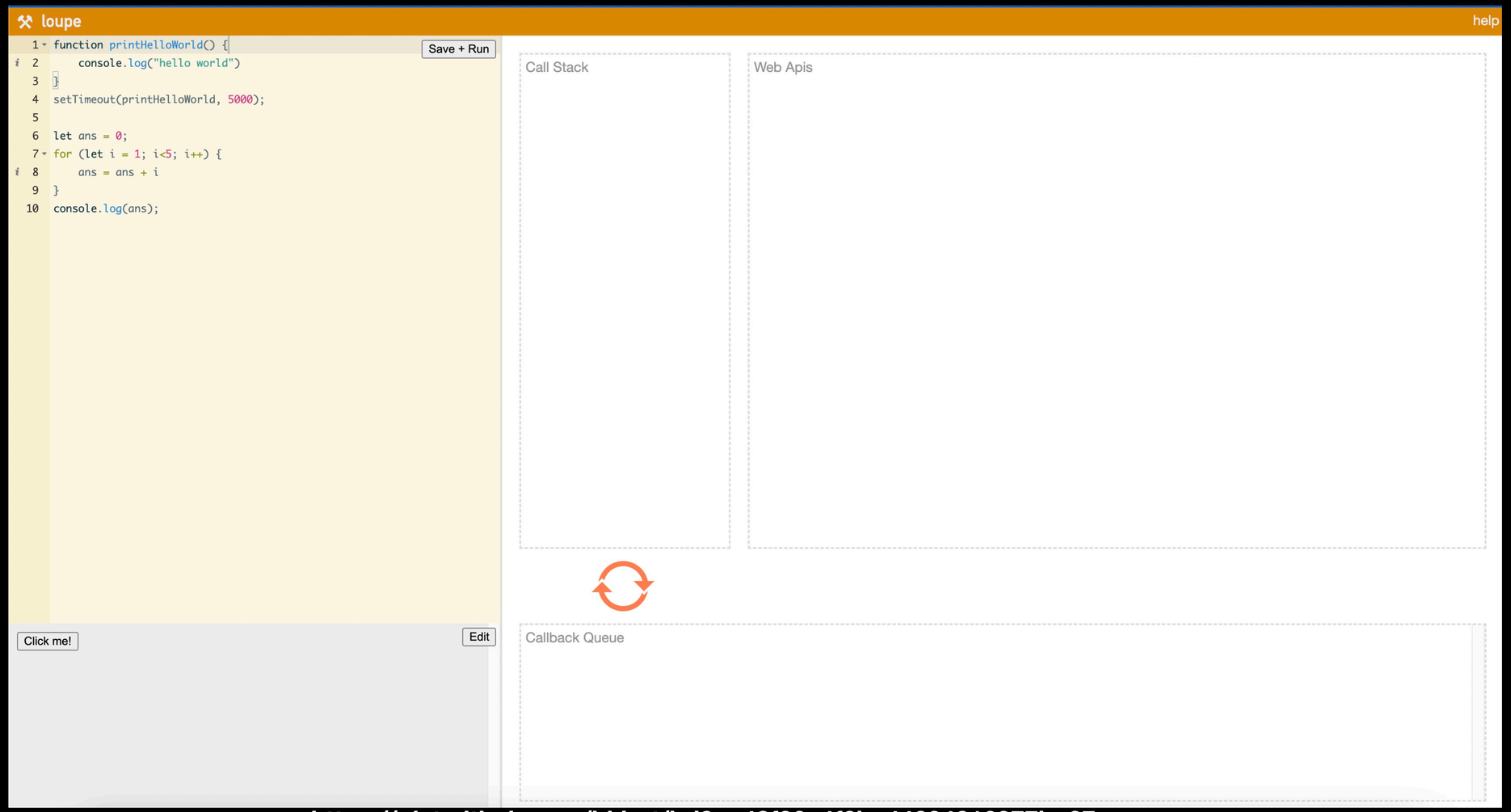


Reads from the filesystem

Async functions vs sync functions

Lets look at the javascript architecture that lets us achieve this asynchronous nature

http://latentflip.com/loupe



https://gist.github.com/hkirat/bd8ea48f26e4f2ba44204216377ba27c

Good checkpoint

Async functions vs sync functions, real use of callbacks ✅

JS Browser architecture ✅

Promises

Async await

Promises

Lets see the communication b/w these 2

Timmy

Simmy

Promises

Lets see the communication b/w these 2

Timmy gives her a task, and a callback function

Timmy

Simmy

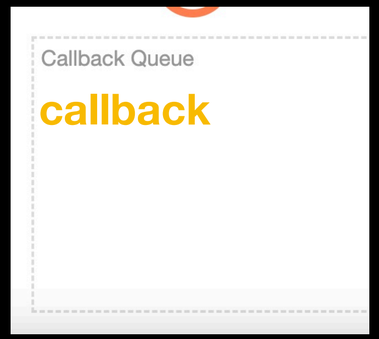
Promises

Lets see the communication b/w these 2

Timmy

Simmy

She does the task (reads from file/waits for 1s ...)



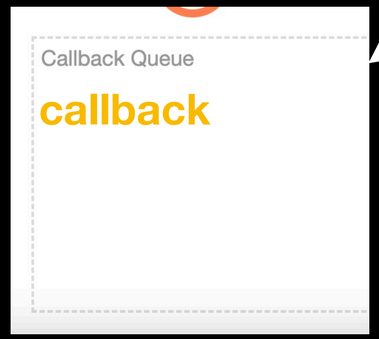
Promises

Lets see the communication b/w these 2

Timmy

Simmy

She puts the result in the callback queue



Promises

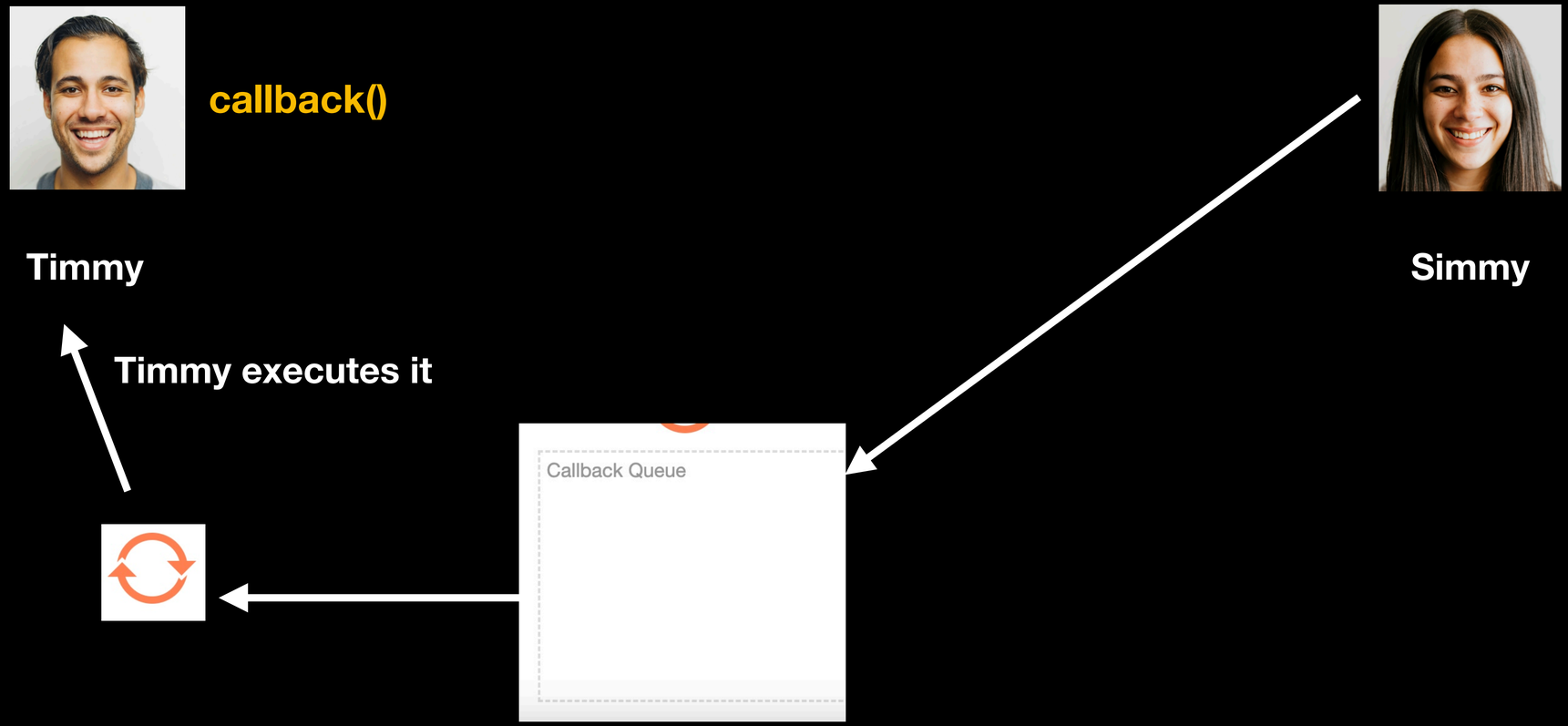
Lets see the communication b/w these 2

Timmy

Simmy

Promises

Lets see the communication b/w these 2



Promises

What does the code look like?

index.js > ...

```
1  function findSum(n) {
2    let ans = 0;
3    for (let i = 0; i<n; i++) {
4      ans += i;
5    }
6    return ans;
7  }
8
9  function findSumTill100() {
10   return findSum(100);
11 }
12
13 setTimeout(findSumTill100, 1000)
14 console.log("hello world");
```

Promises

This code is ugly
Promises are syntactical sugar that make this code slightly more readable

index.js > ...

```
1  function findSum(n) {
2    let ans = 0;
3    for (let i = 0; i<n; i++) {
4      ans += i;
5    }
6    return ans;
7  }
8
9  function findSumTill100() {
10   return findSum(100);
11 }
12
13 setTimeout(findSumTill100, 1000)
14 console.log("hello world");
```

Promises

Until now, we've only used other people's asynchronous functions
How can we create an asynchronous function of our own?

Promises

Until now, we've only used other people's asynchronous functions
How can we create an asynchronous function of our own?

Ugly way

index.js > ...

```
1  const fs = require('fs');
2
3  // my own asynchronous function
4  function kiratsReadFile(cb) {
5    fs.readFile("a.txt", "utf-8", function(err, data) {
6      cb(data);
7    });
8  }
9
10 // callback function to call
11 function onDone(data) {
12   console.log(data)
13 }
14
15 kiratsReadFile(onDone)
```

https://gist.github.com/hkirat/621d9168e39d99bcc14d767c912e9777

It is just a wrapper on top of another async function, which is fine.
Usually all async functions you will write will be on top of JS provided async functions like setTimeout or fs.readFile

Promises

Until now, we've only used other people's asynchronous functions
How can we create an asynchronous function of our own?

1. **Breathe before the next slide**
2. **It's supposed to be overwhelming**
3. **You only need to understand the syntax for now**
4. **Don't worry about how it actually works under the hood**

Promises

Until now, we've only used other people's asynchronous functions
How can we create an asynchronous function of our own?

Cleaner way (promises)

index.js > kiratsReadFile > ...

```
1  const fs = require('fs');
2
3  // my own asynchronous function
4  function kiratsReadFile() {
5    return new Promise(function(resolve) {
6      fs.readFile("a.txt", "utf-8", function(err, data) {
7        resolve(data);
8      });
9    })
10 }
11
12 // callback function to call
13 function onDone(data) {
14   console.log(data)
15 }
16
17 kiratsReadFile().then(onDone);
```

https://gist.github.com/hkirat/75be07d3ea18f93cba907c85233b4217

Its just syntactic sugar
Still uses callbacks under the hood

Lets see how Timmy and Simmy talk

index.js > kiratsReadFile > ...

```
1  const fs = require('fs');
2
3  // my own asynchronous function
4  function kiratsReadFile() {
5    return new Promise(function(resolve) {
6      fs.readFile("a.txt", "utf-8", function(err, data) {
7        resolve(data);
8      });
9    })
10 }
11
12 // callback function to call
13 function onDone(data) {
14   console.log(data)
15 }
16
17 kiratsReadFile().then(onDone);
```

Timmy

Can you read a file, promise me simarpreet

Simmy

Lets see how Timmy and Simmy talk

index.js > kiratsReadFile > ...

```
1  const fs = require('fs');
2
3  // my own asynchronous function
4  function kiratsReadFile() {
5    return new Promise(function(resolve) {
6      fs.readFile("a.txt", "utf-8", function(err, data) {
7        resolve(data);
8      });
9    })
10 }
11
12 // callback function to call
13 function onDone(data) {
14   console.log(data)
15 }
16
17 kiratsReadFile().then(onDone);
```

Timmy

Sure, here's a promise,
I may or may not resolve this promise
Returns it immediately

(Notice, no callback anywhere)

Simmy

Lets see how Timmy and Simmy talk

index.js > kiratsReadFile > ...

```
1  const fs = require('fs');
2
3  // my own asynchronous function
4  function kiratsReadFile() {
5    return new Promise(function(resolve) {
6      fs.readFile("a.txt", "utf-8", function(err, data) {
7        resolve(data);
8      });
9    })
10 }
11
12 // callback function to call
13 function onDone(data) {
14   console.log(data)
15 }
16
17 kiratsReadFile().then(onDone);
```

Timmy

Does her thing, reads the file

Simmy

Lets see how Timmy and Simmy talk

index.js > kiratsReadFile > ...

```
1  const fs = require('fs');
2
3  // my own asynchronous function
4  function kiratsReadFile() {
5    return new Promise(function(resolve) {
6      fs.readFile("a.txt", "utf-8", function(err, data) {
7        resolve(data);
8      });
9    })
10 }
11
12 // callback function to call
13 function onDone(data) {
14   console.log(data)
15 }
16
17 kiratsReadFile().then(onDone);
```

Timmy

Calls resolve with the final data

Simmy

Lets see how Timmy and Simmy talk

index.js > kiratsReadFile > ...

```
1  const fs = require('fs');
2
3  // my own asynchronous function
4  function kiratsReadFile() {
5    return new Promise(function(resolve) {
6      fs.readFile("a.txt", "utf-8", function(err, data) {
7        resolve(data);
8      });
9    })
10 }
11
12 // callback function to call
13 function onDone(data) {
14   console.log(data)
15 }
16
17 kiratsReadFile().then(onDone);
```

onDone gets called on Timmy's side

Timmy

Simmy

Lets see how Timmy and Simmy talk

Ugly code

index.js > ...

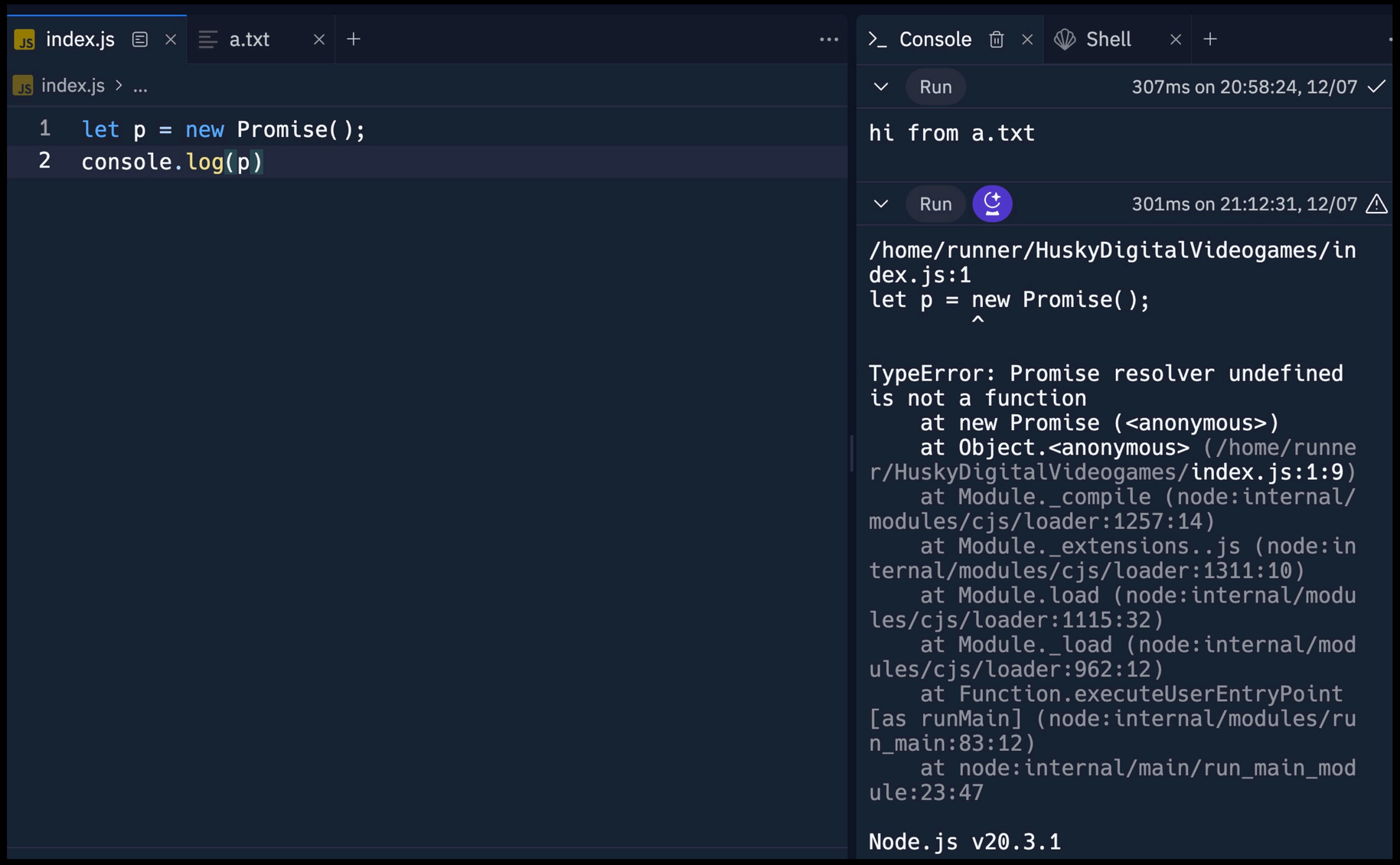
```
1  const fs = require('fs');
2
3  // my own asynchronous function
4  function kiratsReadFile(cb) {
5    fs.readFile("a.txt", "utf-8", function(err, data) {
6      cb(data);
7    });
8  }
9
10 // callback function to call
11 function onDone(data) {
12   console.log(data)
13 }
14
15 kiratsReadFile(onDone)
```

Pretty code

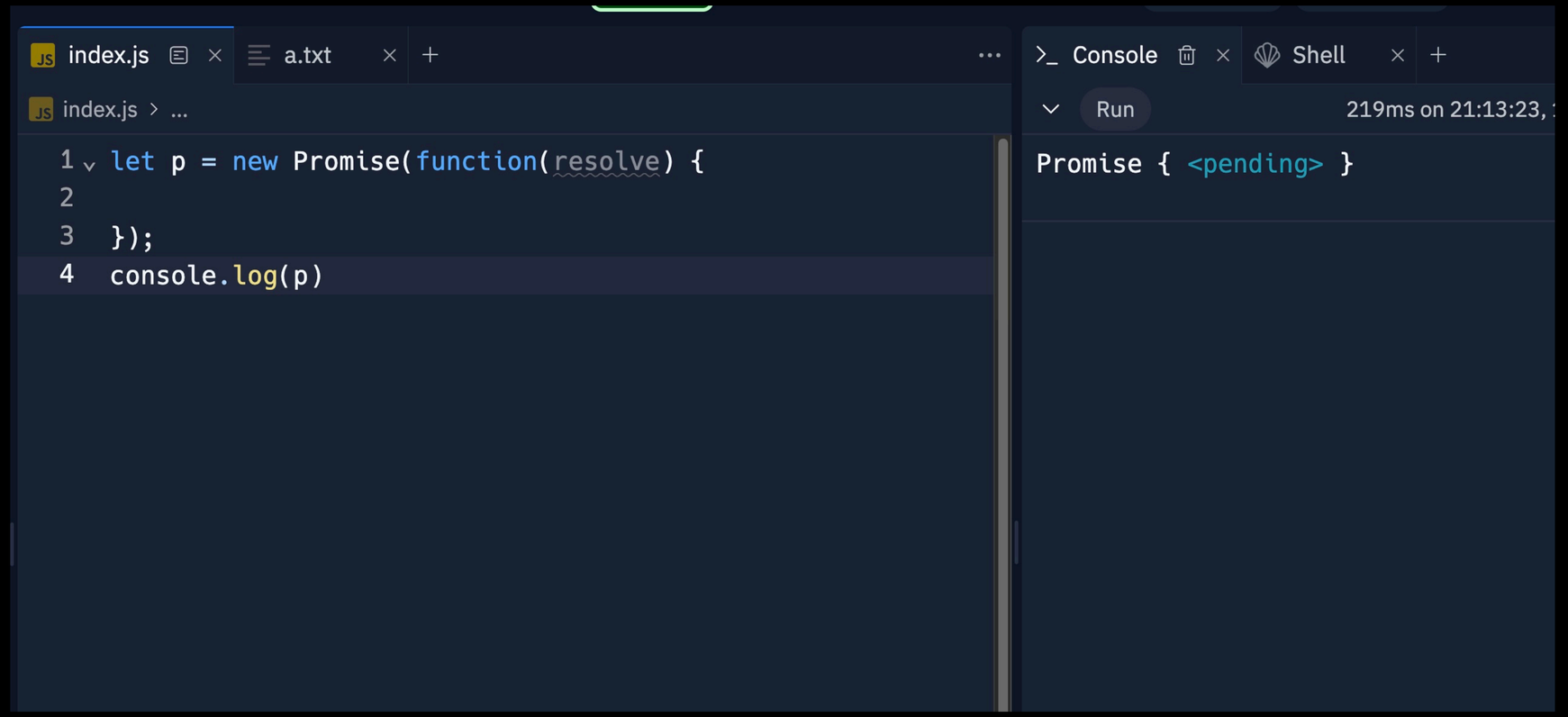
index.js > kiratsReadFile > ...

```
1  const fs = require('fs');
2
3  // my own asynchronous function
4  function kiratsReadFile() {
5    return new Promise(function(resolve) {
6      fs.readFile("a.txt", "utf-8", function(err, data) {
7        resolve(data);
8      });
9    })
10 }
11
12 // callback function to call
13 function onDone(data) {
14   console.log(data)
15 }
16
17 kiratsReadFile().then(onDone);
```

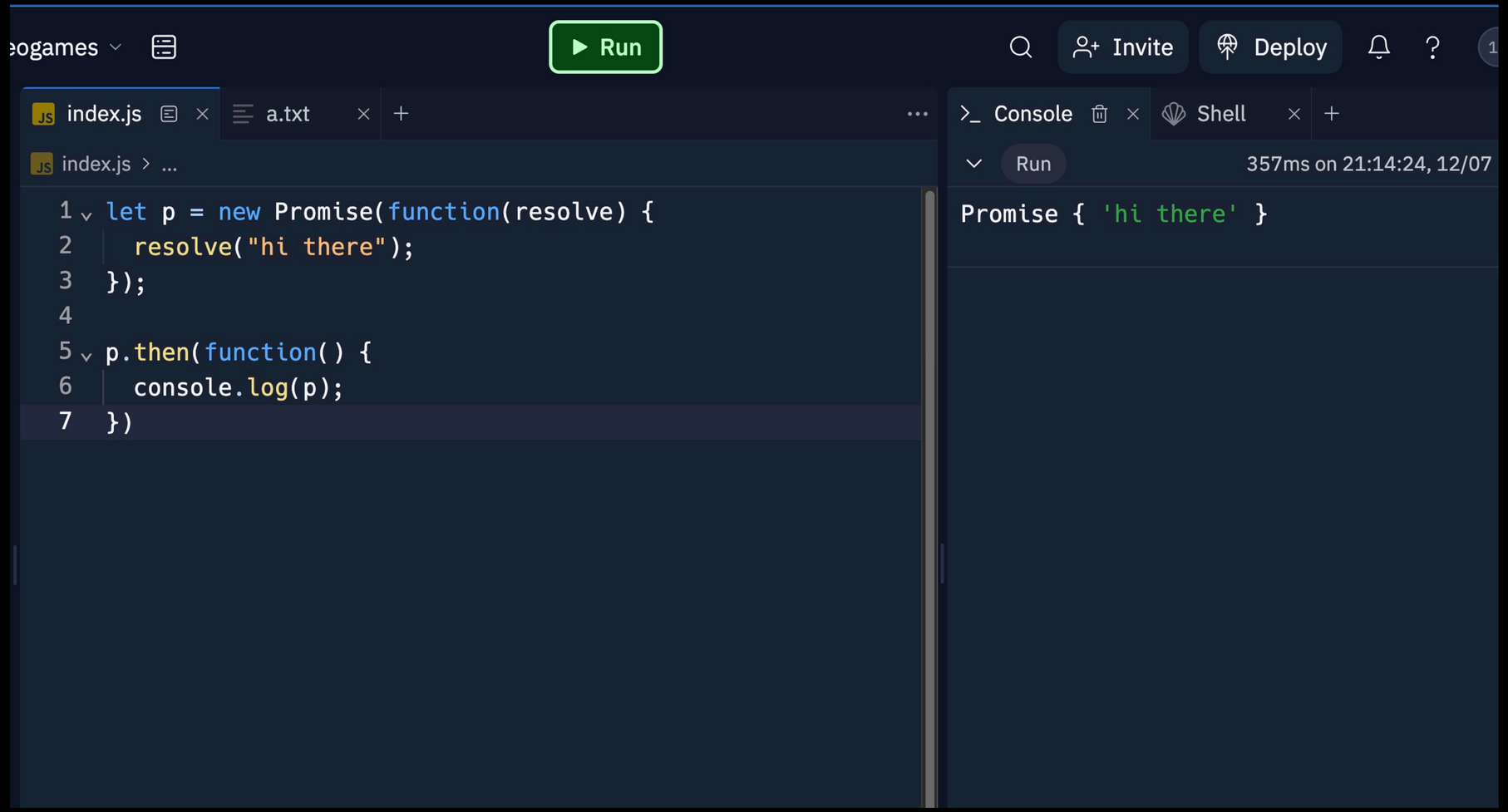
What even is a promise?
It is just a class that makes callbacks and async functions slightly more readable



What even is a promise?
Whenever u create it, you need to pass in a function as the first argument which has resolve as the First argument

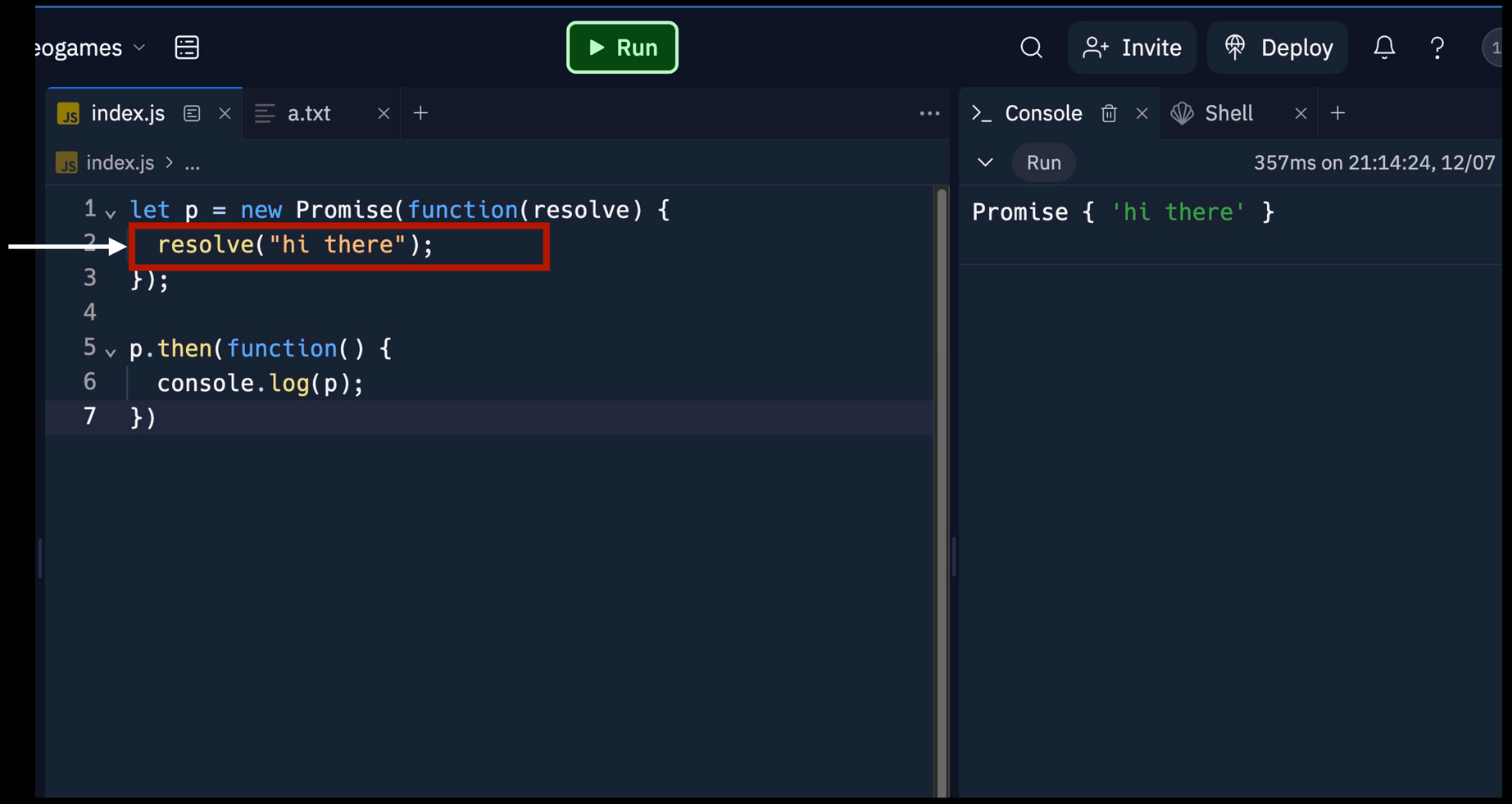


Here’s a simple promise that immediately resolves



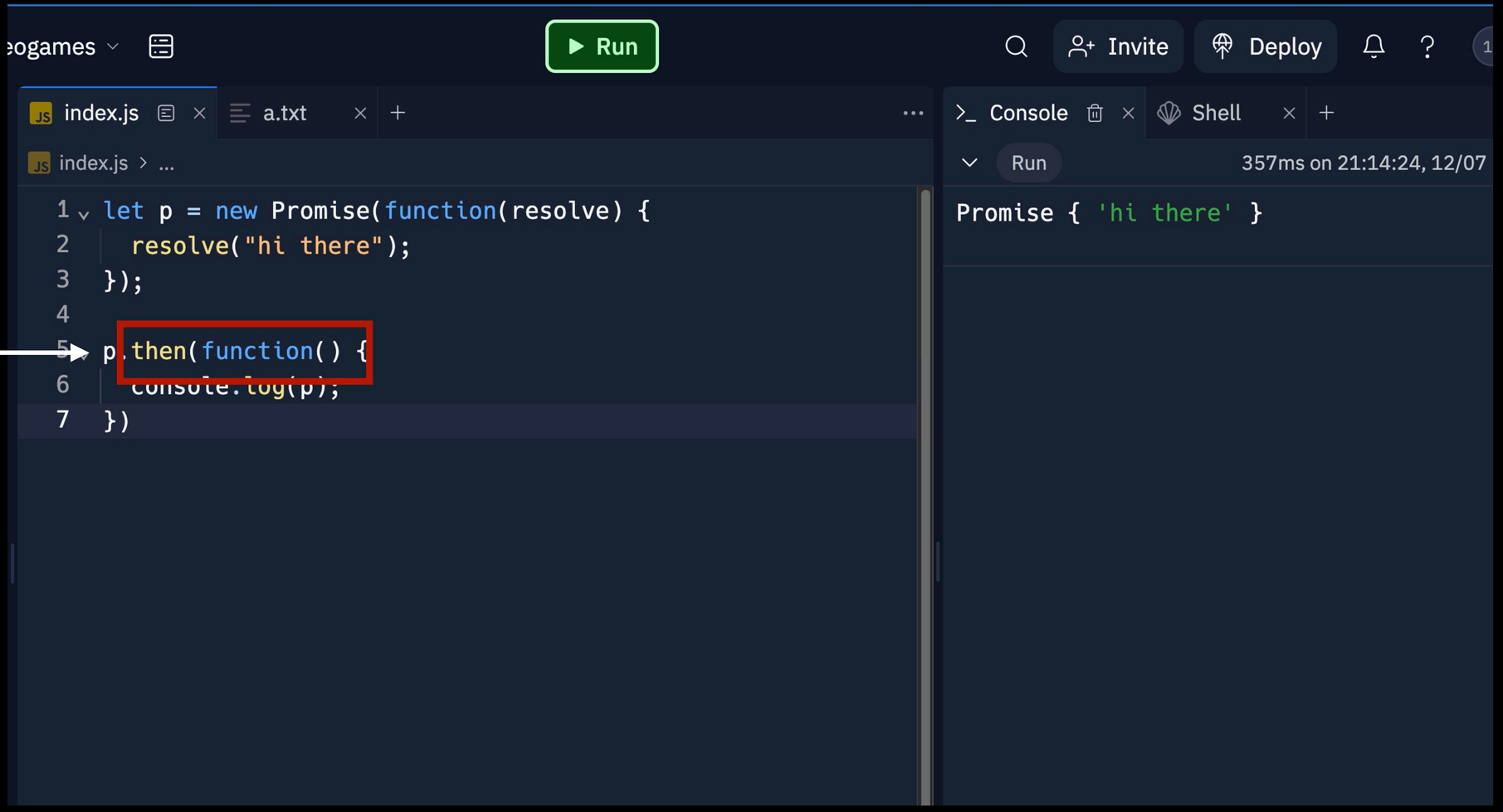
Promises

Place for the writer of the async function to do their magic (get ketchup) and call resolve at the end with the data



Promises

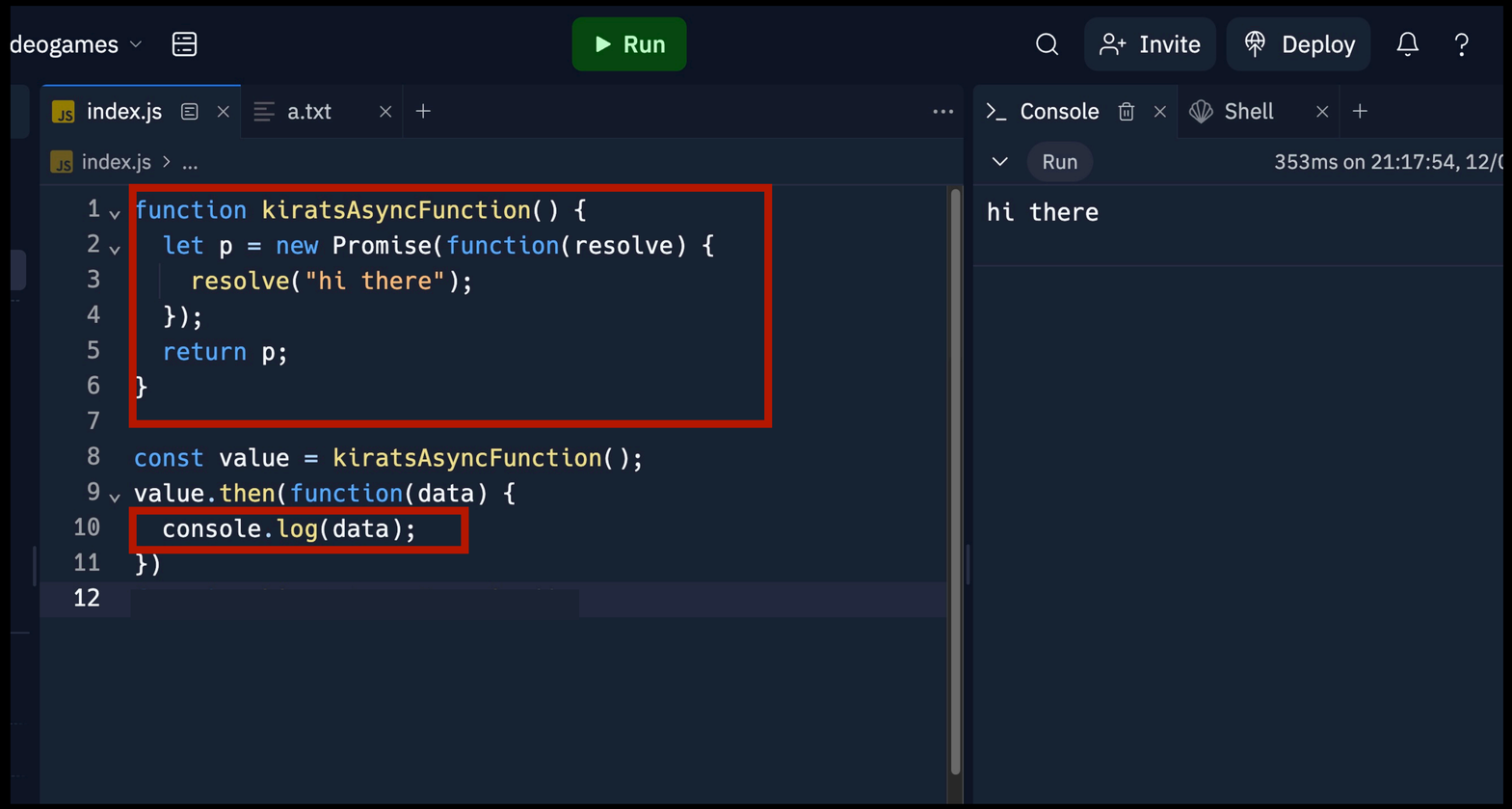
.then gets called whenever the async function resolves



Promises

Dummy async function that almost immediately resolves

Actually logging the data with what the function above Resolved with

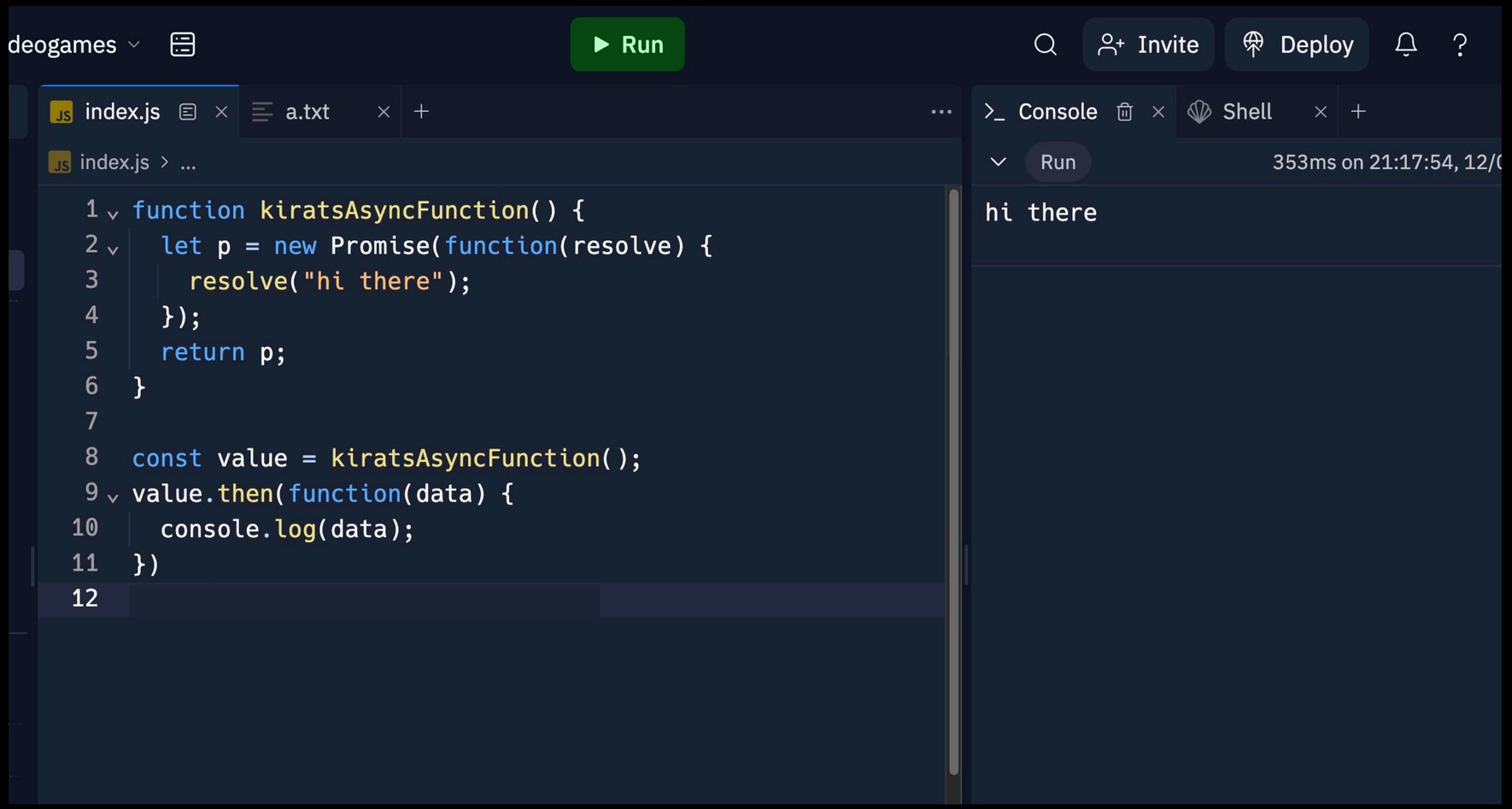


https://gist.github.com/hkirat/46580244f43ca2847cfd57664a3803b1

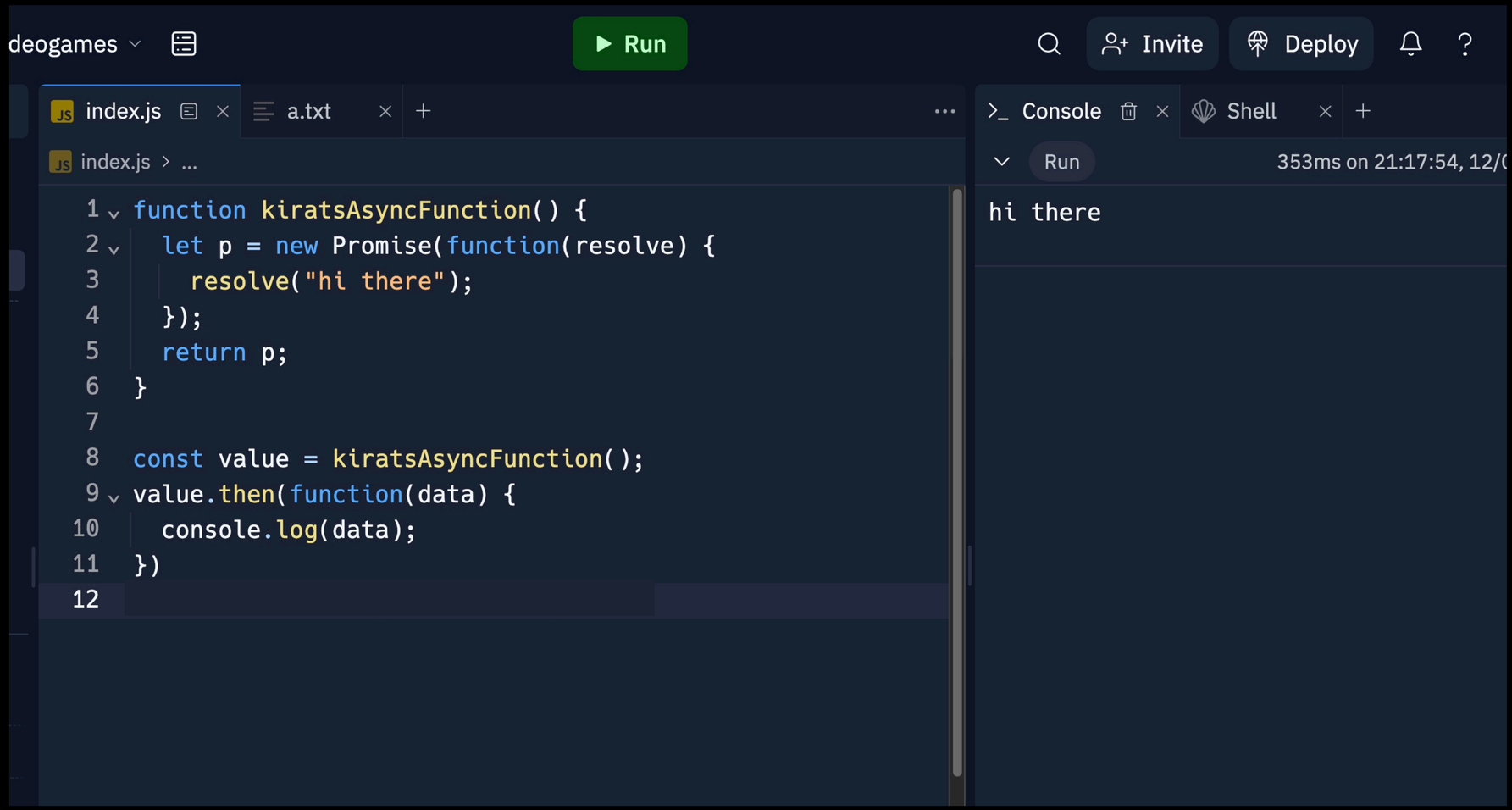
Promises

Try to marinate both the sides

1. Creating an async fn
2. Calling an async fn



Why even make it async Harkirat seems like you are just doing something that can be done with a normal function



https://gist.github.com/hkirat/46580244f43ca2847cfd57664a3803b1

Why even make it async Harkirat seems like you are just doing something that can be done with a normal function

Intimidating async function

index.js > ...

```
1  function kiratsAsyncFunction() {
2    let p = new Promise(function(resolve) {
3      resolve("hi there");
4    });
5    return p;
6  }
7
8  const value = kiratsAsyncFunction();
9  value.then(function(data) {
10   console.log(data);
11 })
12
```

Simple function

index.js | a.txt

index.js > ...

```
1  function kiratsAsyncFunction() {
2    return "hi there"
3  }
4
5  const data = kiratsAsyncFunction();
6  console.log(data);
```

You actually can, but you will need to pass in a callback which is what promises help you write in a cleaner fashion

Intimidating async function

index.js a.txt

index.js > ...

```
1  function kiratsAsyncFunction() {
2    let p = new Promise(function(resolve) {
3      setTimeout(resolve, 2000)
4    });
5    return p;
6  }
7
8  const value = kiratsAsyncFunction();
9  value.then(function() {
10   console.log("hi there");
11 })
12
```

Simple function

index.js a.txt

index.js > ...

```
1  function kiratsAsyncFunction(callback) {
2    setTimeout(callback, 2000);
3  }
4
5  kiratsAsyncFunction(function() {
6   console.log("hello!");
7  });
8
9
```

Async functions vs sync functions, real use of callbacks

JS Browser architecture

Promises

Async await

Async await

Aynsc await syntax
(Again, just syntactic sugar. Still uses callbacks/Promises under the hood)

Normal syntax

index.js a.txt

index.js > ...

```
1  function kiratsAsyncFunction() {
2    let p = new Promise(function(resolve) {
3      // do some async logic here
4      resolve("hi there!")
5    });
6    return p;
7  }
8
9  function main() {
10   kiratsAsyncFunction().then(function(value) {
11       console.log(value);
12   });
13 }
14
15 main();
```

https://gist.github.com/hkirat/ceac2c9198f1411ba8f5aea3ada769f3

Async/await syntax

index.js a.txt

index.js > ...

```
1  function kiratsAsyncFunction() {
2    let p = new Promise(function(resolve) {
3      // do some async logic here
4      resolve("hi there!")
5    });
6    return p;
7  }
8
9  async function main() {
10   const value = await kiratsAsyncFunction();
11   console.log(value);
12 }
13
14 main();
```

https://gist.github.com/hkirat/9e00de2335d1ead90436adef68c9b2f9

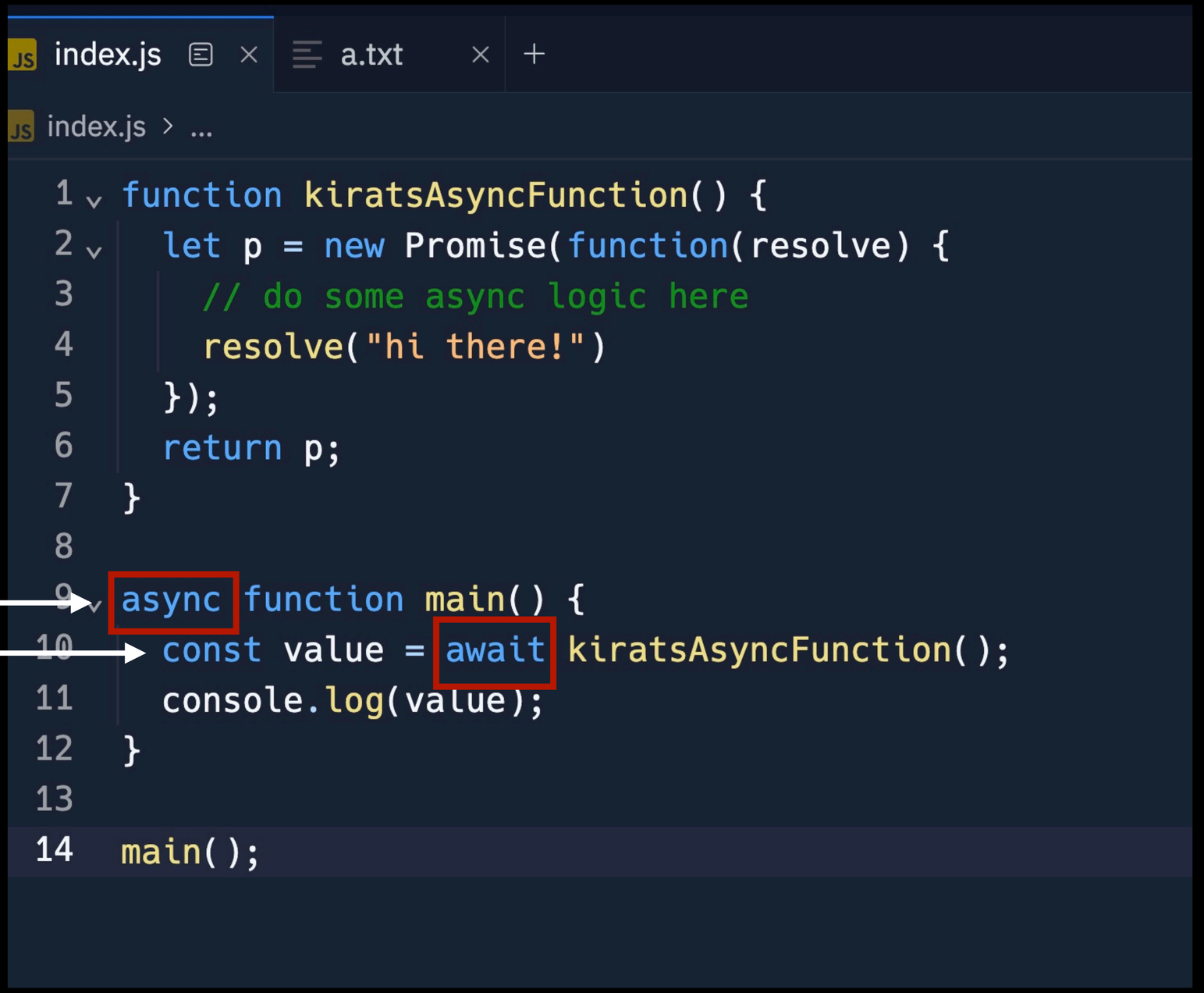
Aynsc await syntax

Again, it is just syntactic sugar. Still uses callbacks/Promises under the hood
Makes code much more readable than callbacks/Promises
Usually used on the caller side, not on the side where you define an async function

Aynsc await syntax

Any function that wants to use await, needs to be async

Rather than using the .then syntax, we add await before and get the final value in the variable



Aynsc await syntax
Again, both do the same thing

Normal syntax

index.js a.txt

index.js > ...

```
1  function kiratsAsyncFunction() {
2    let p = new Promise(function(resolve) {
3      // do some async logic here
4      resolve("hi there!")
5    });
6    return p;
7  }
8
9  function main() {
10   kiratsAsyncFunction().then(function(value) {
11       console.log(value);
12   });
13 }
14
15 main();
```

https://gist.github.com/hkirat/ceac2c9198f1411ba8f5aea3ada769f3

Async/await syntax

index.js a.txt

index.js > ...

```
1  function kiratsAsyncFunction() {
2    let p = new Promise(function(resolve) {
3      // do some async logic here
4      resolve("hi there!")
5    });
6    return p;
7  }
8
9  async function main() {
10   const value = await kiratsAsyncFunction();
11   console.log(value);
12 }
13
14 main();
```

https://gist.github.com/hkirat/9e00de2335d1ead90436adef68c9b2f9

Aynsc await syntax
In fact, all three are very similar (becomes more manageable as you move to the right

Callback syntax

index.js > main > ...

```
1  function kiratsAsyncFunction(callback) {
2      // do some async logic here
3      callback("hi there!")
4  }
5
6  async function main() {
7    kiratsAsyncFunction(function(value) {
8      console.log(value);
9    });
10 }
11
12 main();
```

https://gist.github.com/hkirat/9843537dfcaf48ea69df1ec4c4d9091d

Promise (then) syntax

index.js a.txt

index.js > ...

```
1  function kiratsAsyncFunction() {
2    let p = new Promise(function(resolve) {
3      // do some async logic here
4      resolve("hi there!")
5    });
6    return p;
7  }
8
9  function main() {
10   kiratsAsyncFunction().then(function(value) {
11       console.log(value);
12   });
13 }
14
15 main();
```

Async/await syntax

index.js a.txt

index.js > ...

```
1  function kiratsAsyncFunction() {
2    let p = new Promise(function(resolve) {
3      // do some async logic here
4      resolve("hi there!")
5    });
6    return p;
7  }
8
9  async function main() {
10   const value = await kiratsAsyncFunction();
11   console.log(value);
12 }
13
14 main();
```

https://gist.github.com/hkirat/9e00de2335d1ead90436adef68c9b2f9

https://gist.github.com/hkirat/ceac2c9198f1411ba8f5aea3ada769f3