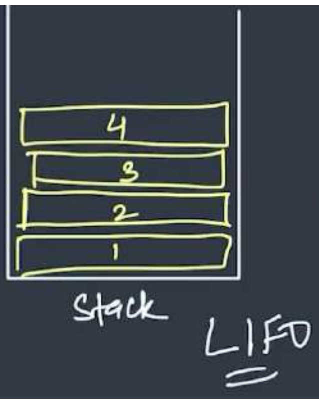
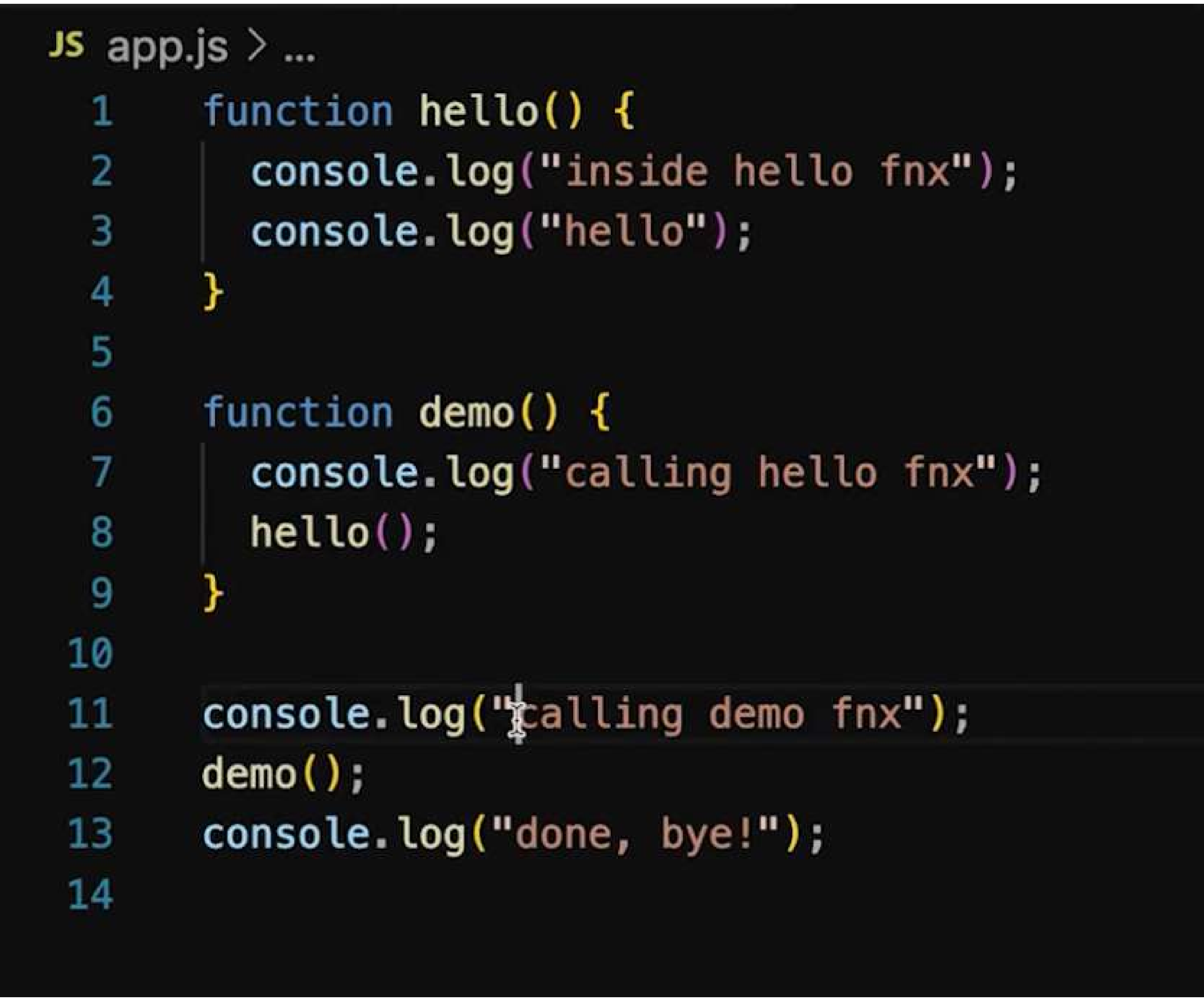
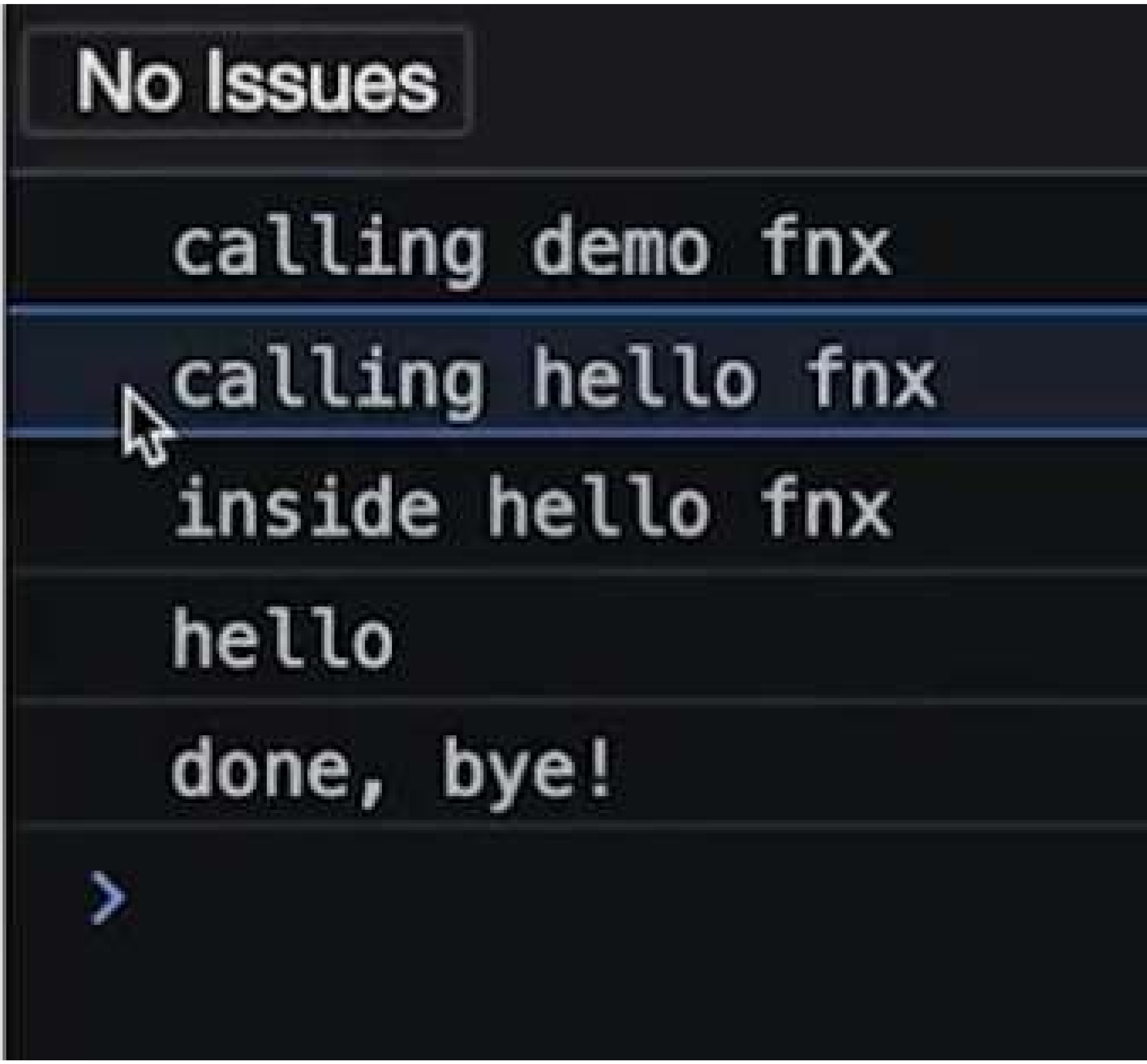
Call Stack

Last In
First Out





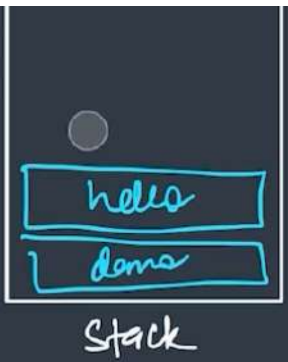
```
function hello() {
  console.log("inside hello fnx");
  console.log("hello");
}

function demo() {
  console.log("calling hello fnx");
  hello();
}

console.log("calling demo fnx");
demo();
console.log("done, bye!");
```



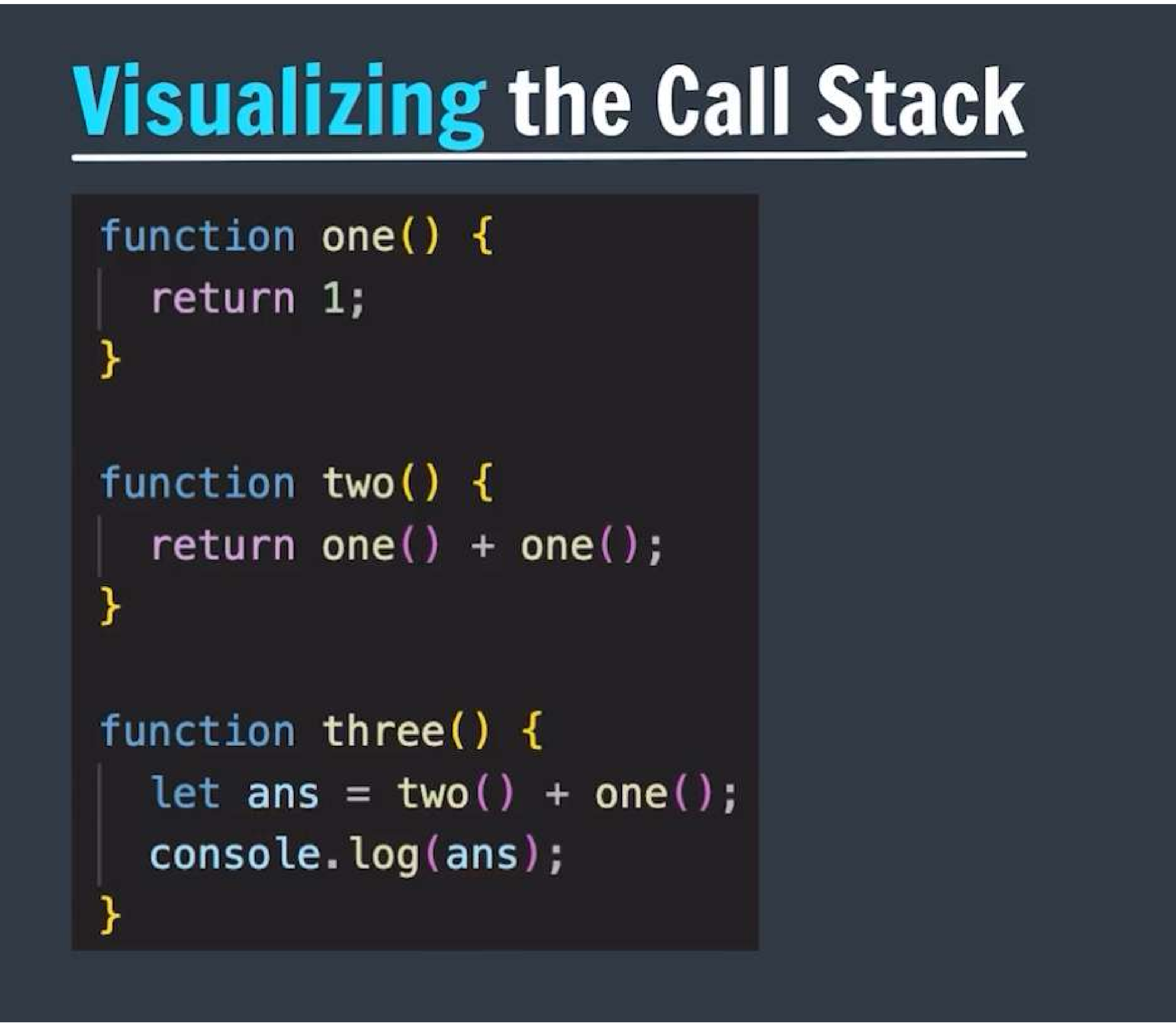
```
calling demo fnx
calling hello fnx
inside hello fnx
hello
done, bye!
>
```

Call Stack

```
hello ( ) {


}


demo( )  {
   hello ( )


}
demo ( );  (1)
```

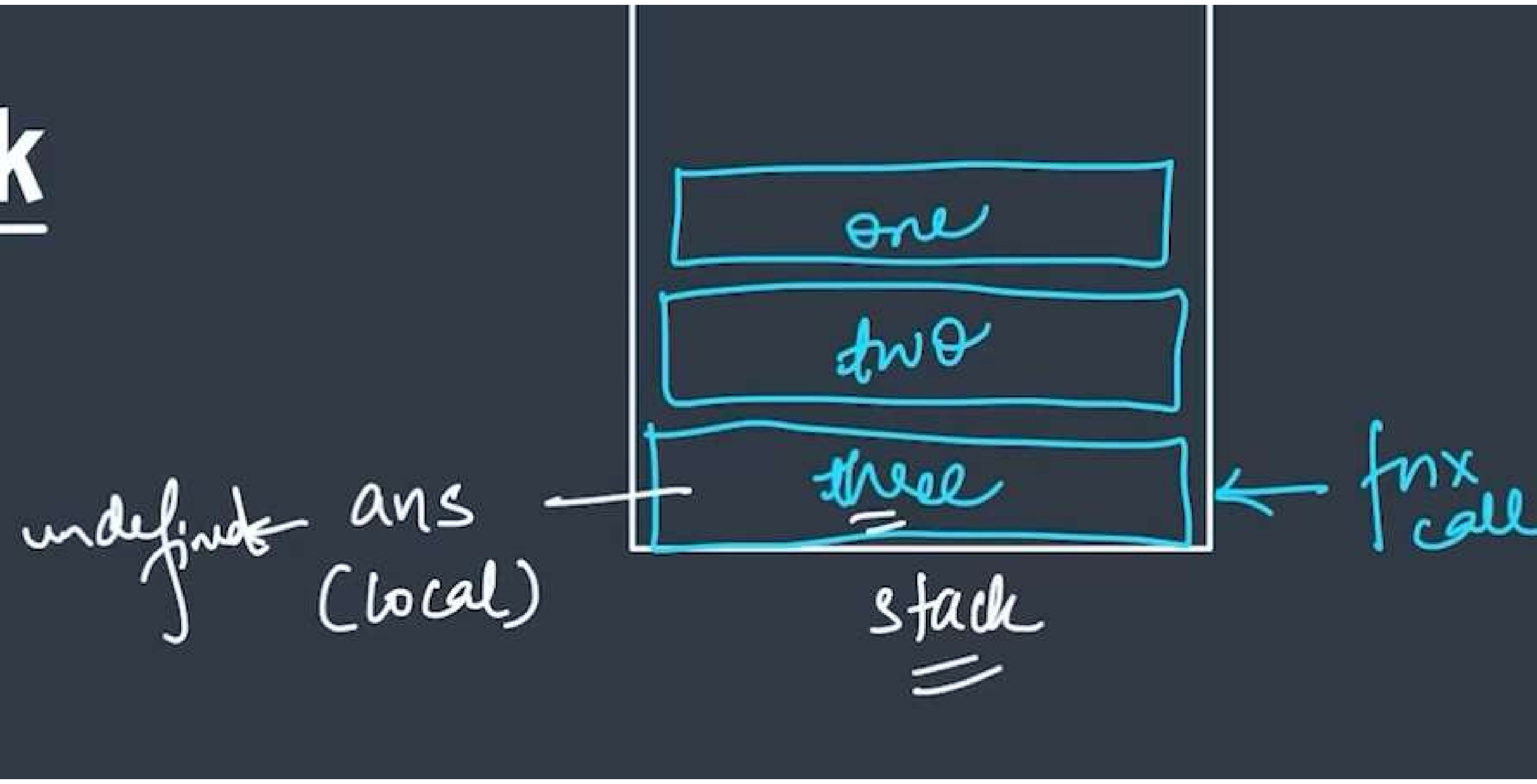


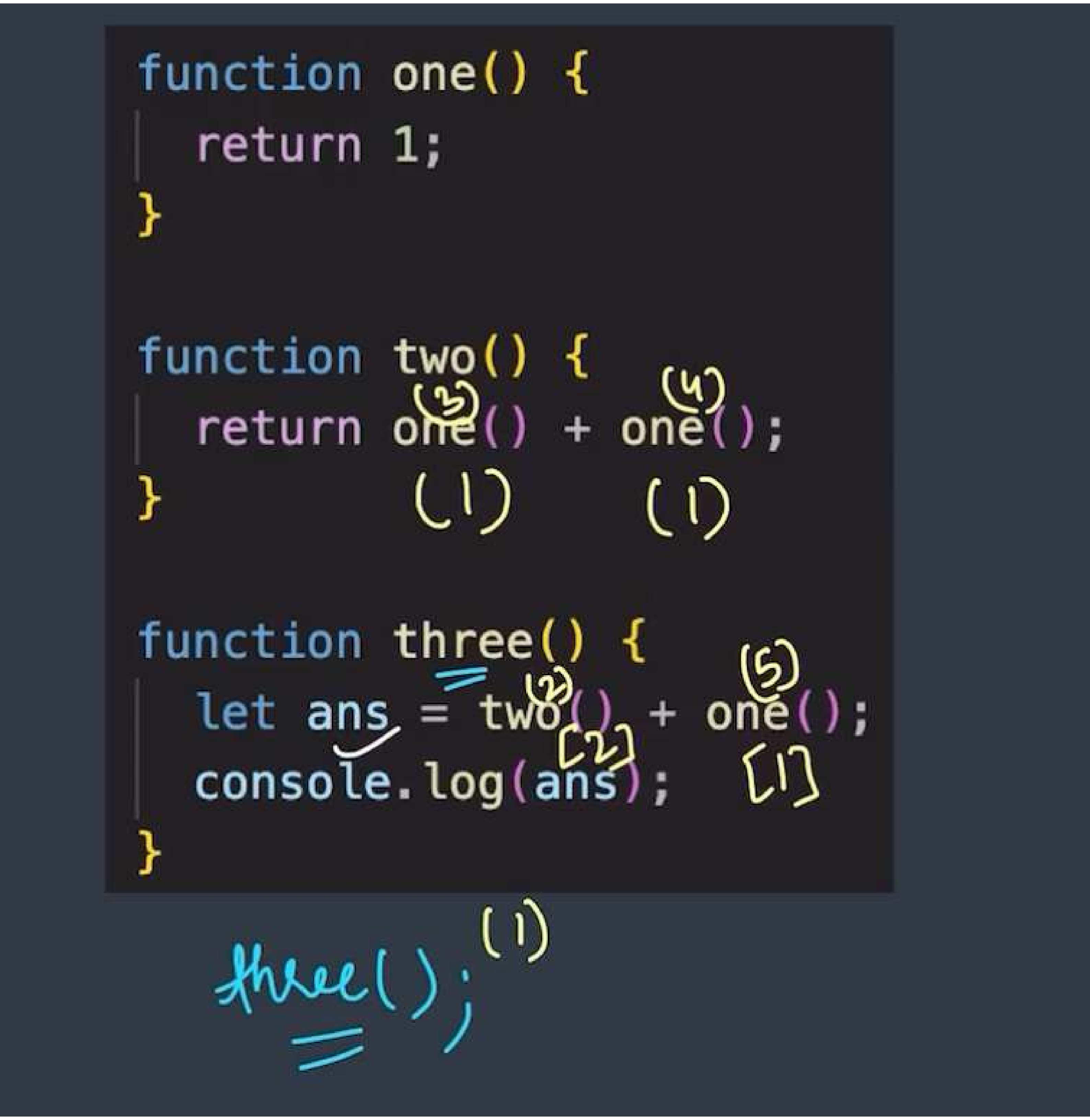
fu
c



Visualizing the Call Stack

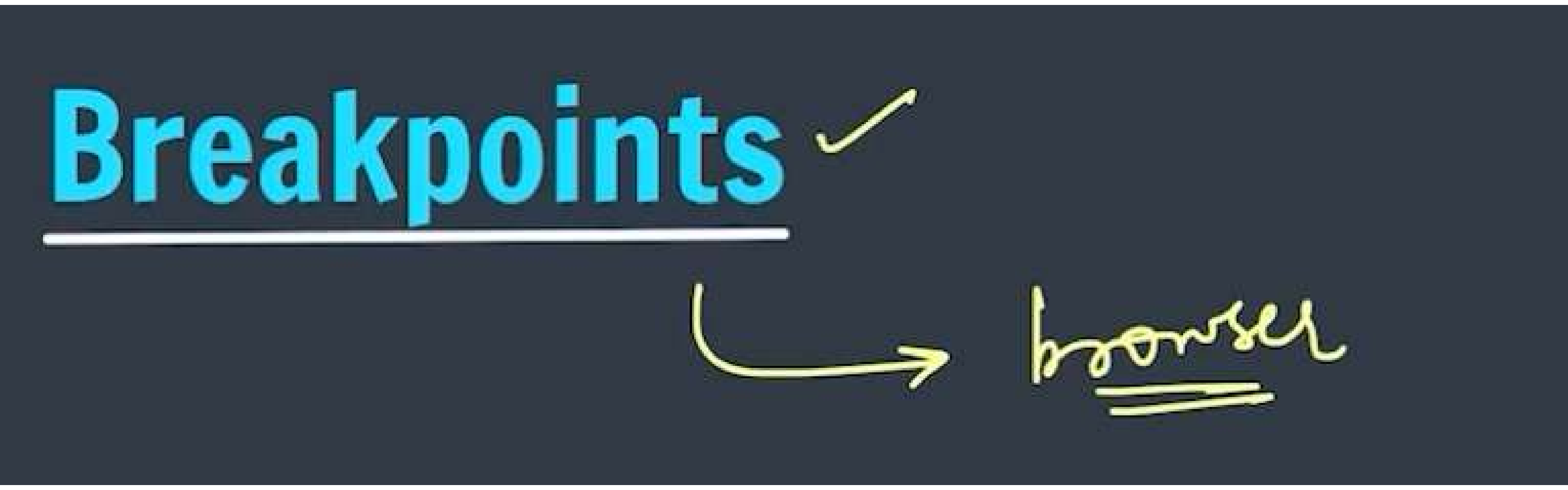
```javascript
function one() {
  return 1;
}

function two() {
  return one() + one();
}

function three() {
  let ans = two() + one();
  console.log(ans);
}
```


k
one
two
three
undefined
ans
(local)
fnx
call
stack



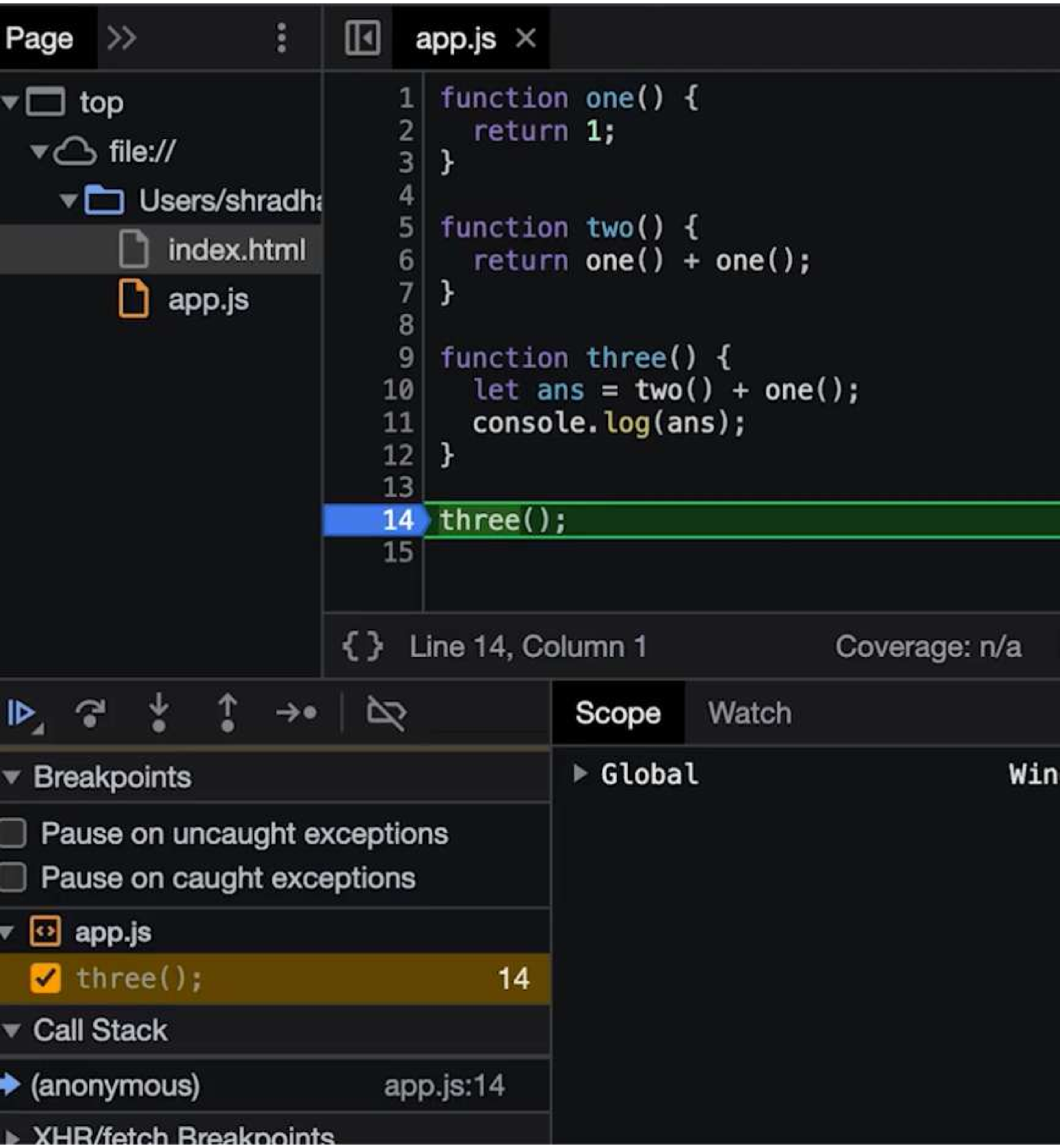
```
function one() {
  return 1;
}

function two() {
  return one() + one();
}

function three() {
  let ans = two() + one();
  console.log(ans);
}
```

three();

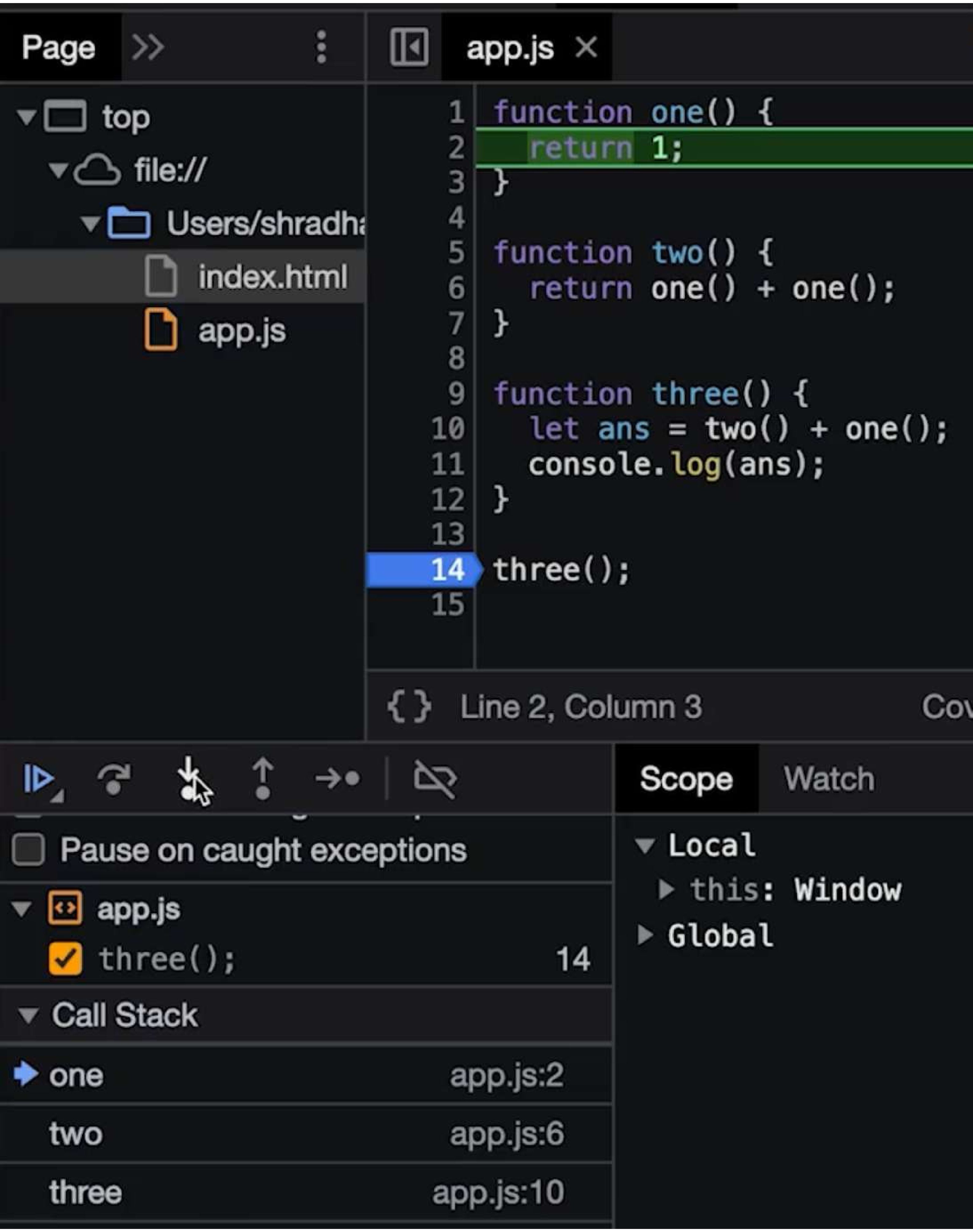


Breakpoints ✓

$\hookrightarrow$ browser



```
function one() {
  return 1;
}

function two() {
  return one() + one();
}

function three() {
  let ans = two() + one();
  console.log(ans);
}

three();
```



Page

top
file://
Users/shradh
index.html
app.js

app.js

```
function one() {
  return 1;
}

function two() {
  return one() + one();
}

function three() {
  let ans = two() + one();
  console.log(ans);
}

three();
```

Line 2, Column 3

Pause on caught exceptions

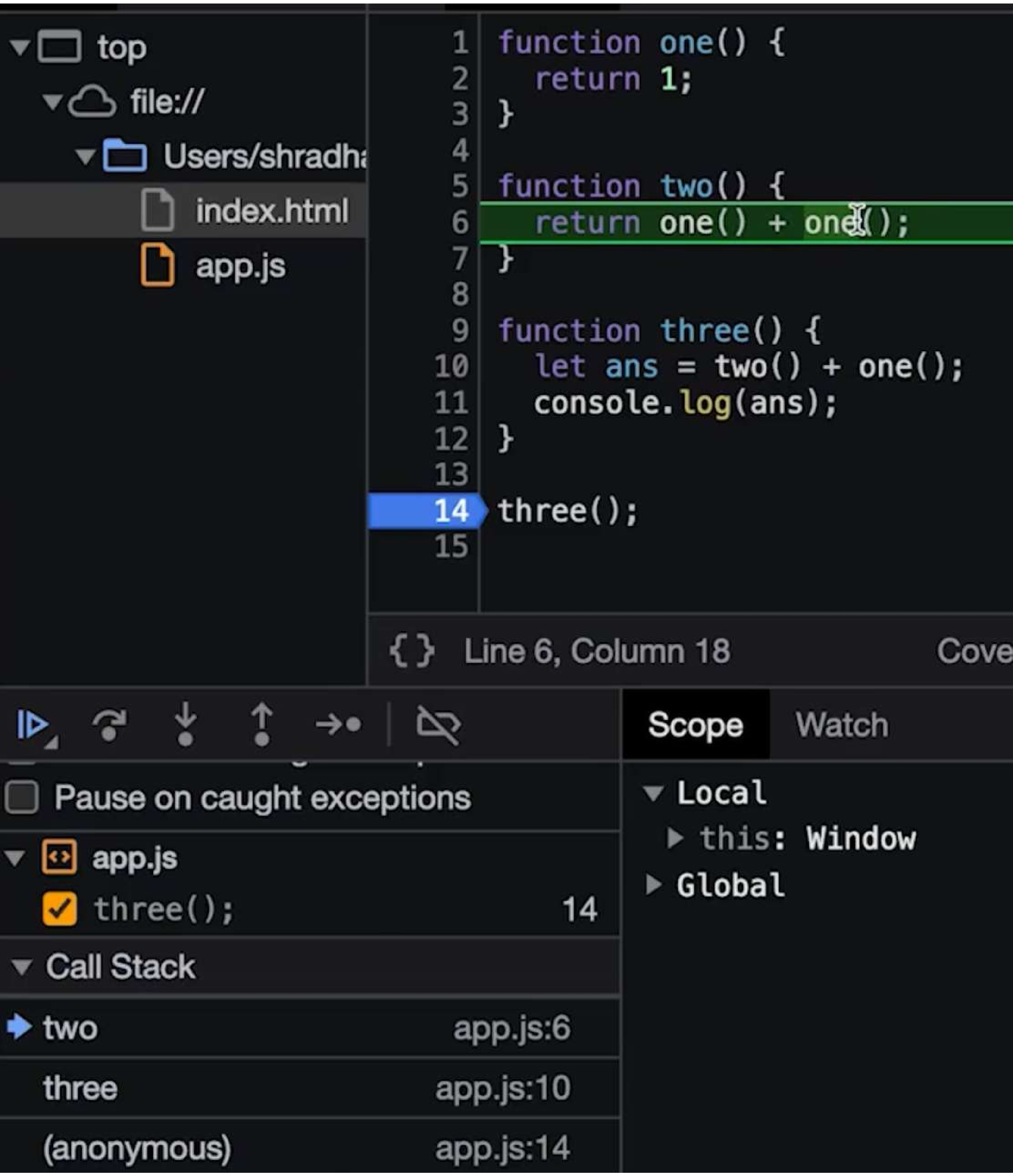
app.js
✓ three(); 14

Call Stack

one	app.js:2
two	app.js:6
three	app.js:10

Scope Watch

```
▼ Local
  ▶ this: Window
▶ Global
```



top
file://
Users/shradha
index.html
app.js

```
function one() {
  return 1;
}

function two() {
  return one() + one();
}

function three() {
  let ans = two() + one();
  console.log(ans);
}

three();

```

Line 6, Column 18

Pause on caught exceptions

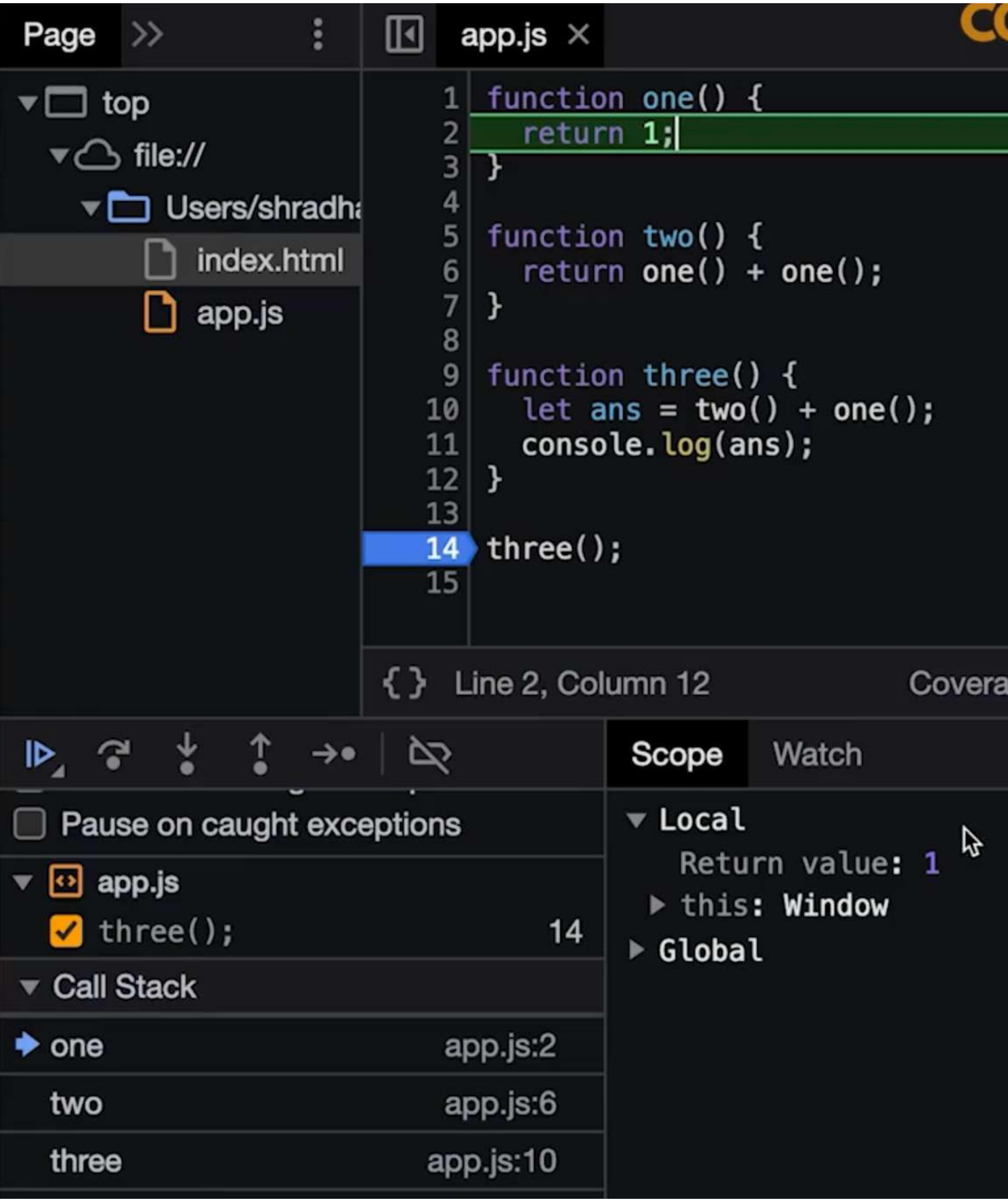
app.js
✓ three(); 14

Call Stack

```
two              app.js:6
three            app.js:10
(anonymous)      app.js:14
```

Scope Watch

```
Local
  this: Window
Global
```



Page

top

file://

Users/shradha

index.html

app.js

```
function one() {
  return 1;
}

function two() {
  return one() + one();
}

function three() {
  let ans = two() + one();
  console.log(ans);
}

three();
```

Line 2, Column 12

Pause on caught exceptions

app.js

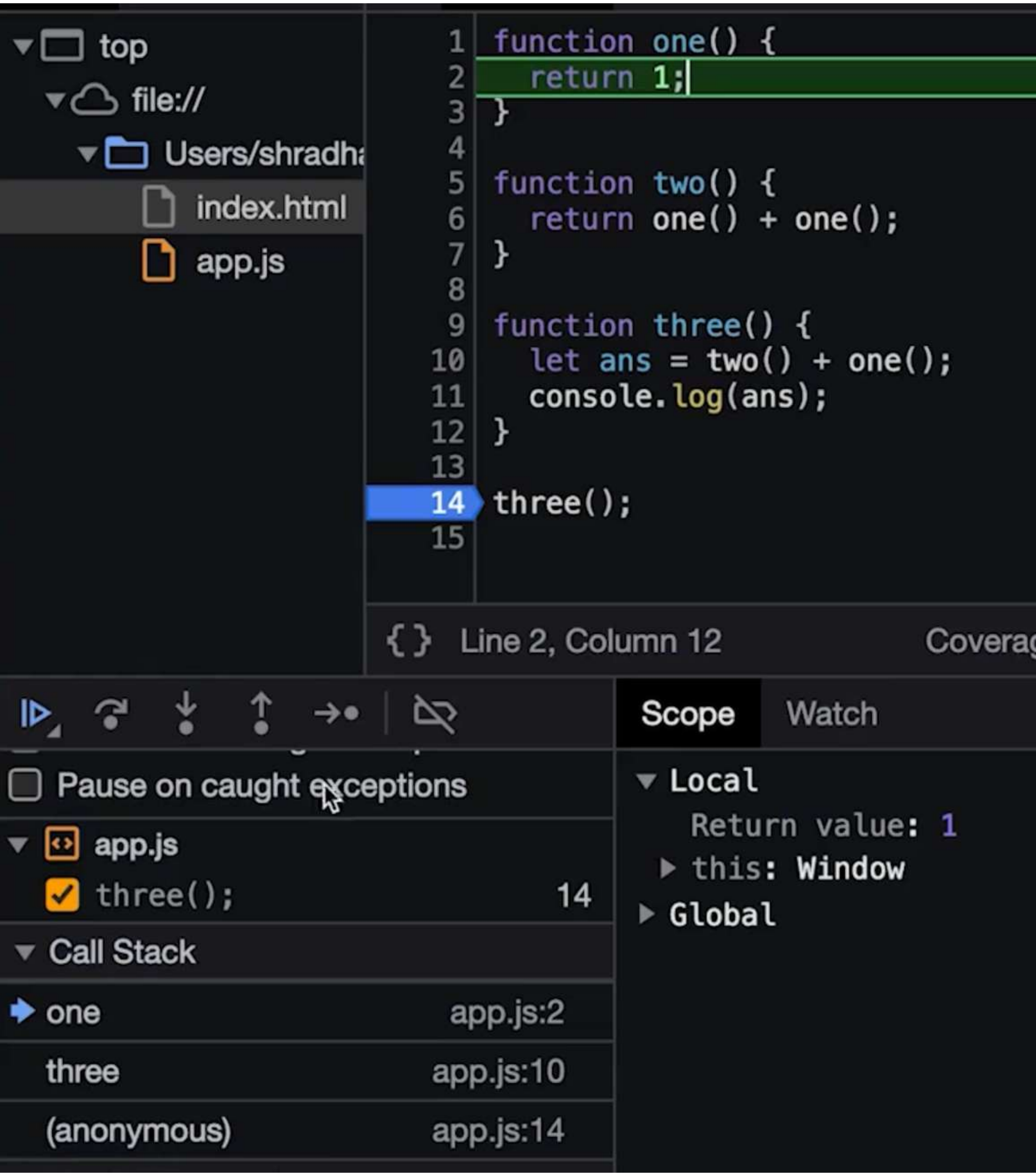
three(); 14

Call Stack

```
one      app.js:2
two      app.js:6
three    app.js:10
```

Scope Watch

```
Local
  Return value: 1
  this: Window
Global
```



```
function one() {
  return 1;
}

function two() {
  return one() + one();
}

function three() {
  let ans = two() + one();
  console.log(ans);
}

three();
```

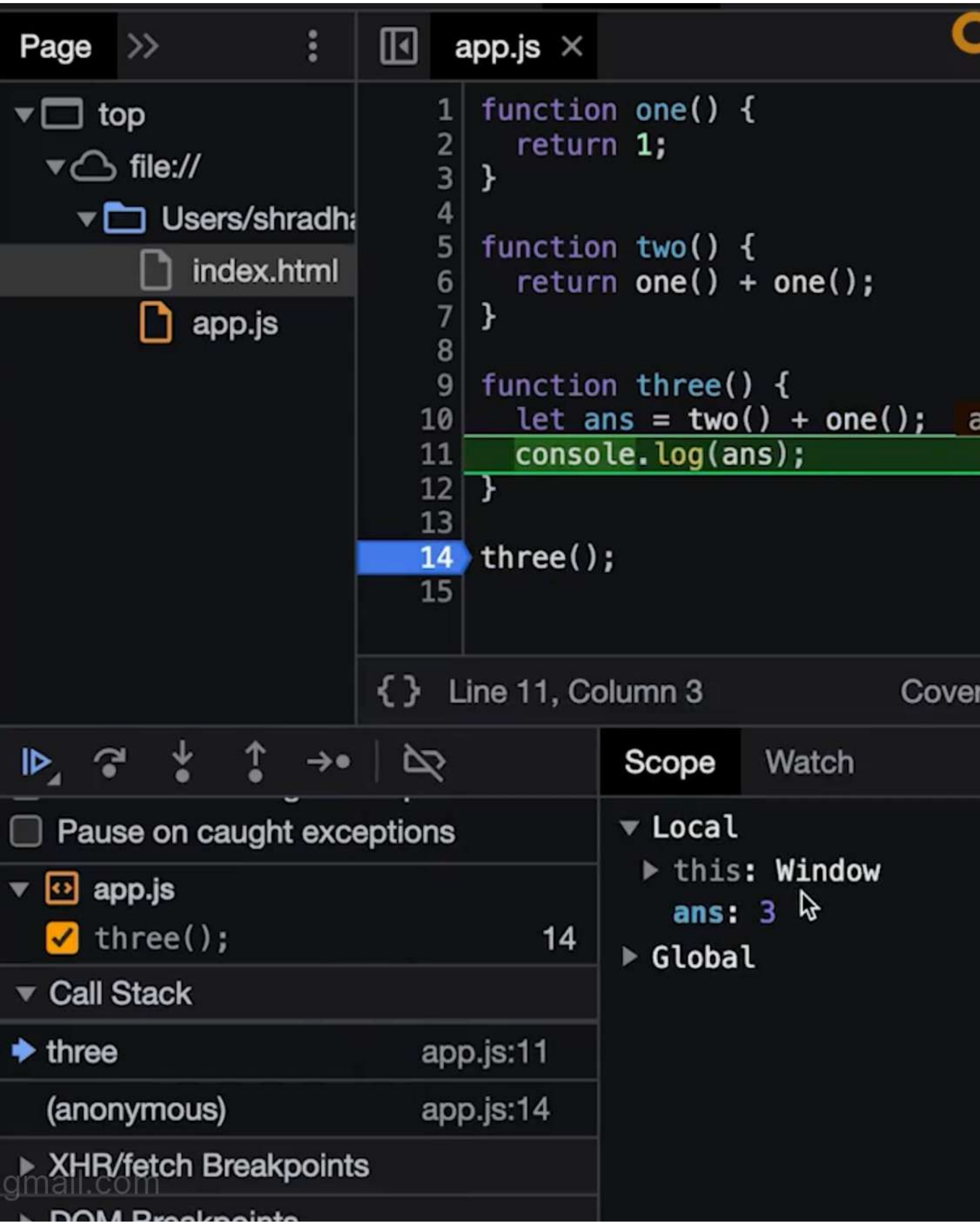
Pause on caught exceptions

app.js
three(); 14

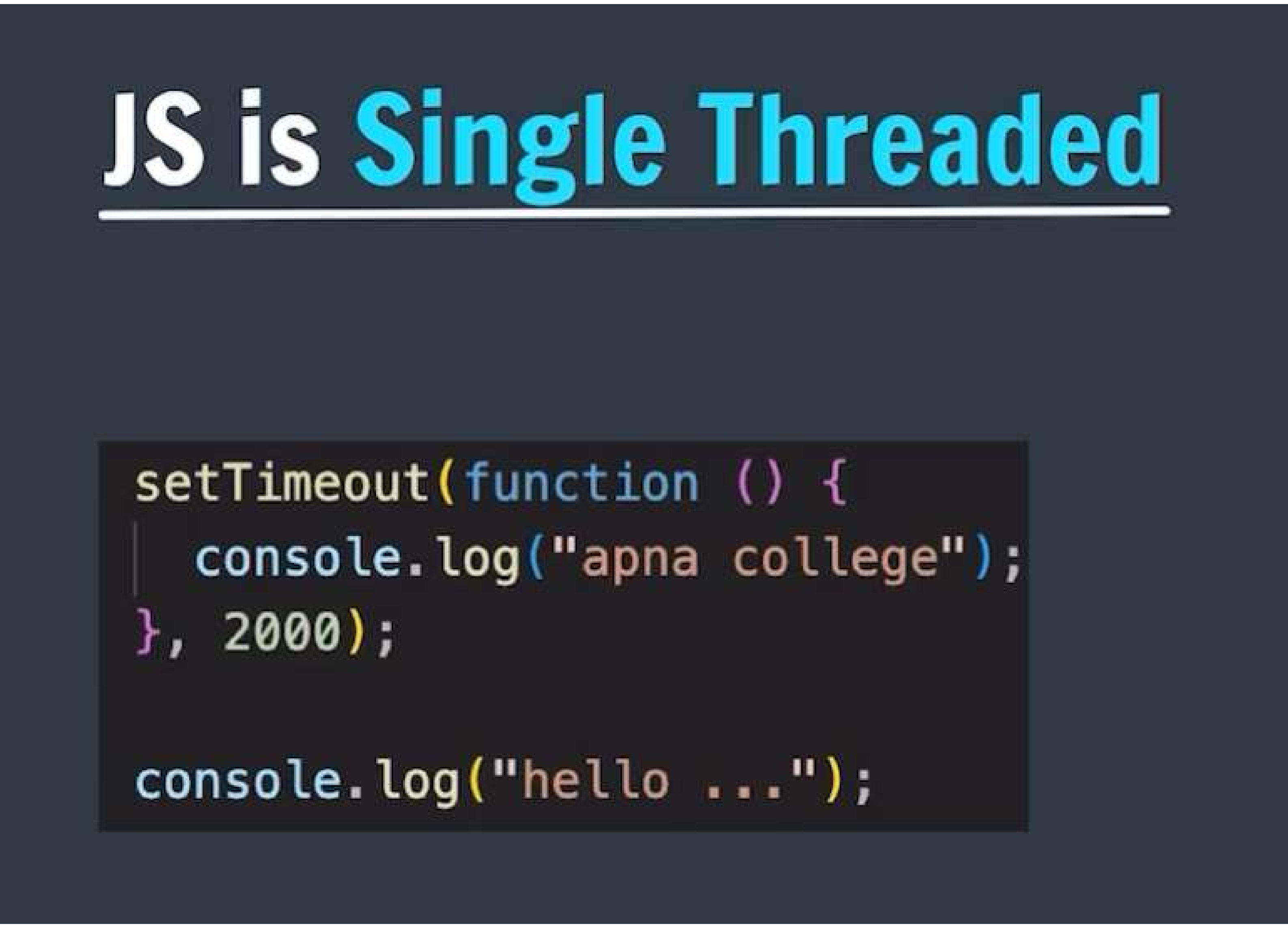
Call Stack

```
one          app.js:2
three        app.js:10
(anonymous)  app.js:14
```

```
Local
  Return value: 1
  this: Window
Global
```

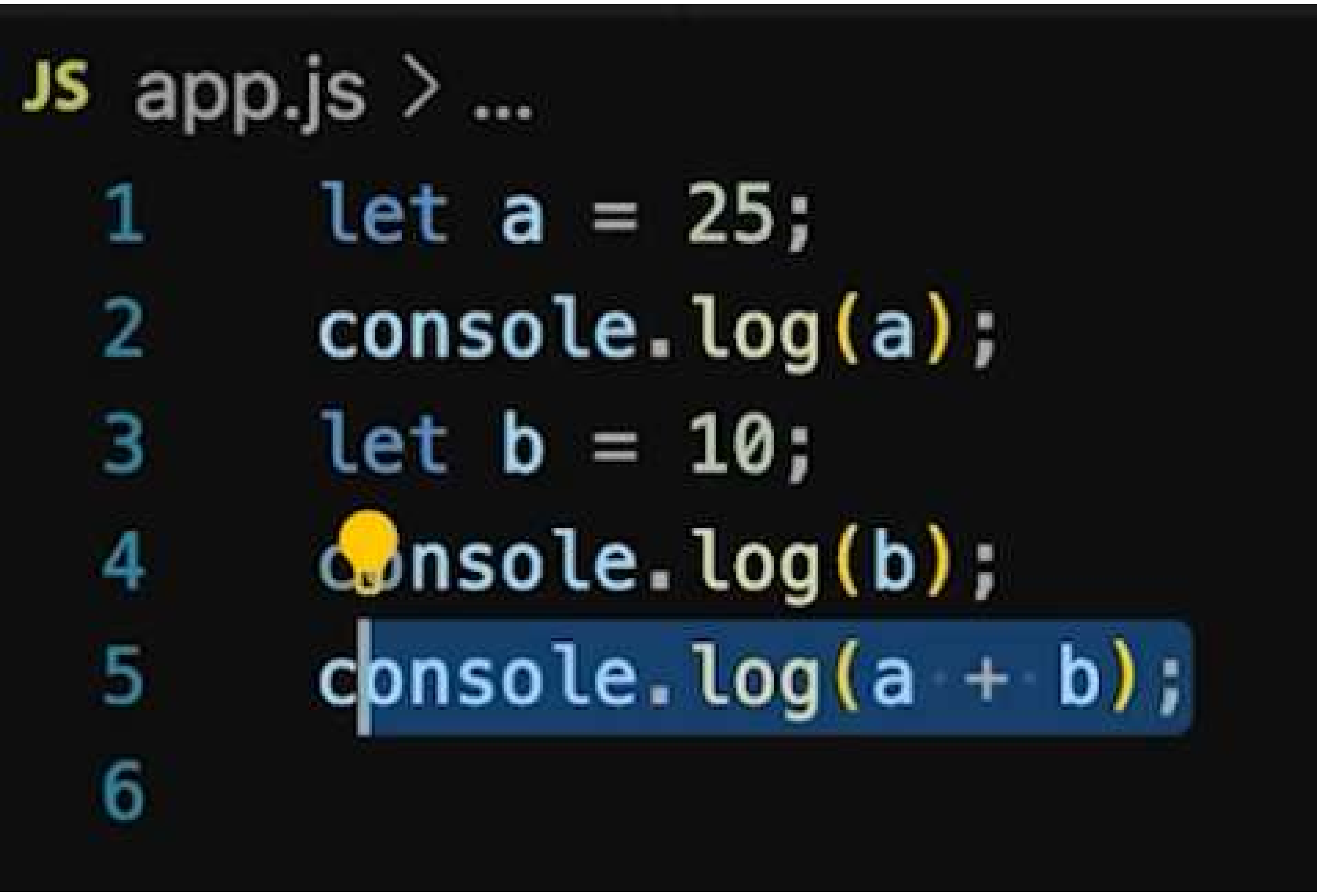


```
function one() {
  return 1;
}

function two() {
  return one() + one();
}

function three() {
  let ans = two() + one();
  console.log(ans);
}

three();
```



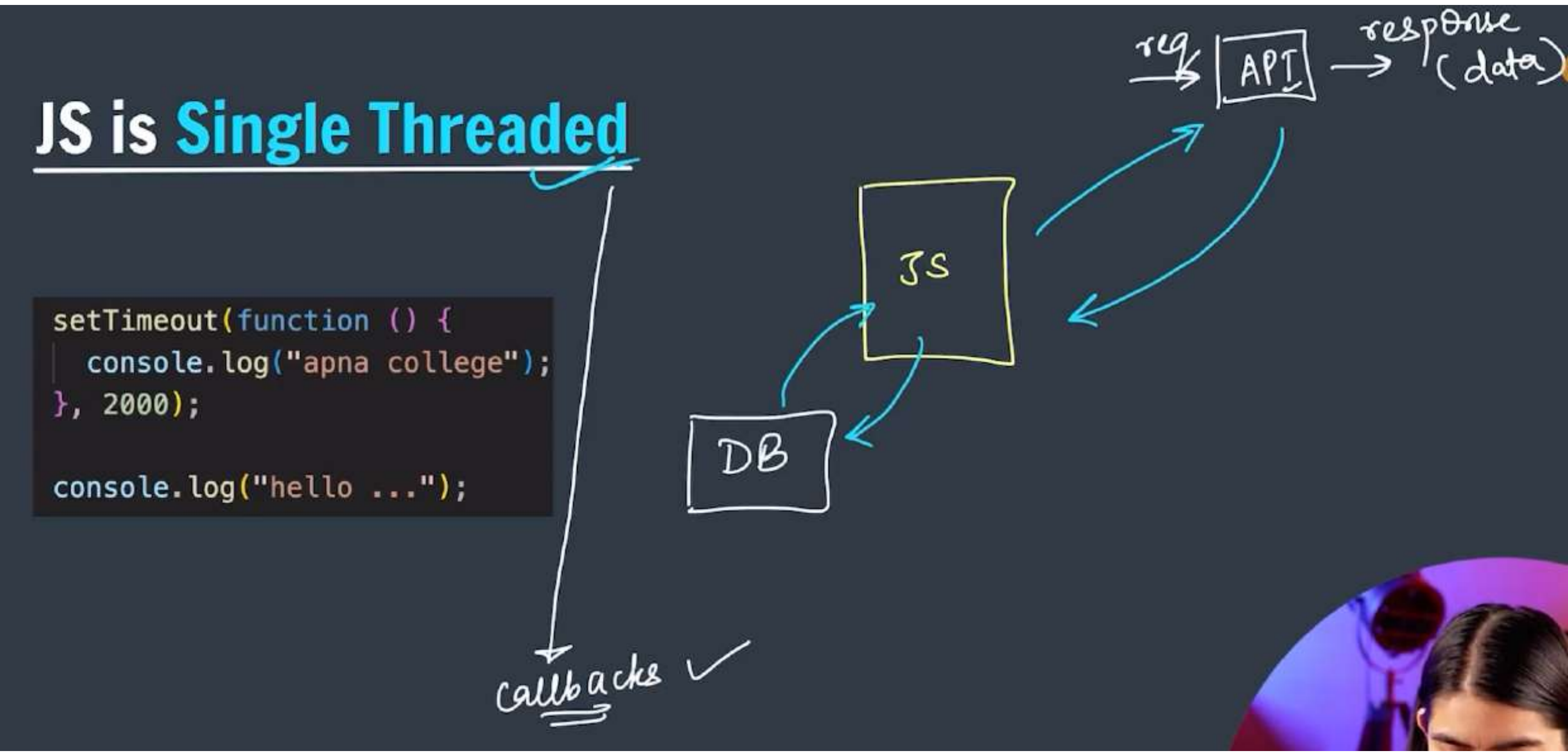
JS is Single Threaded

```
setTimeout(function () {
  console.log("apna college");
}, 2000);

console.log("hello ...");
```



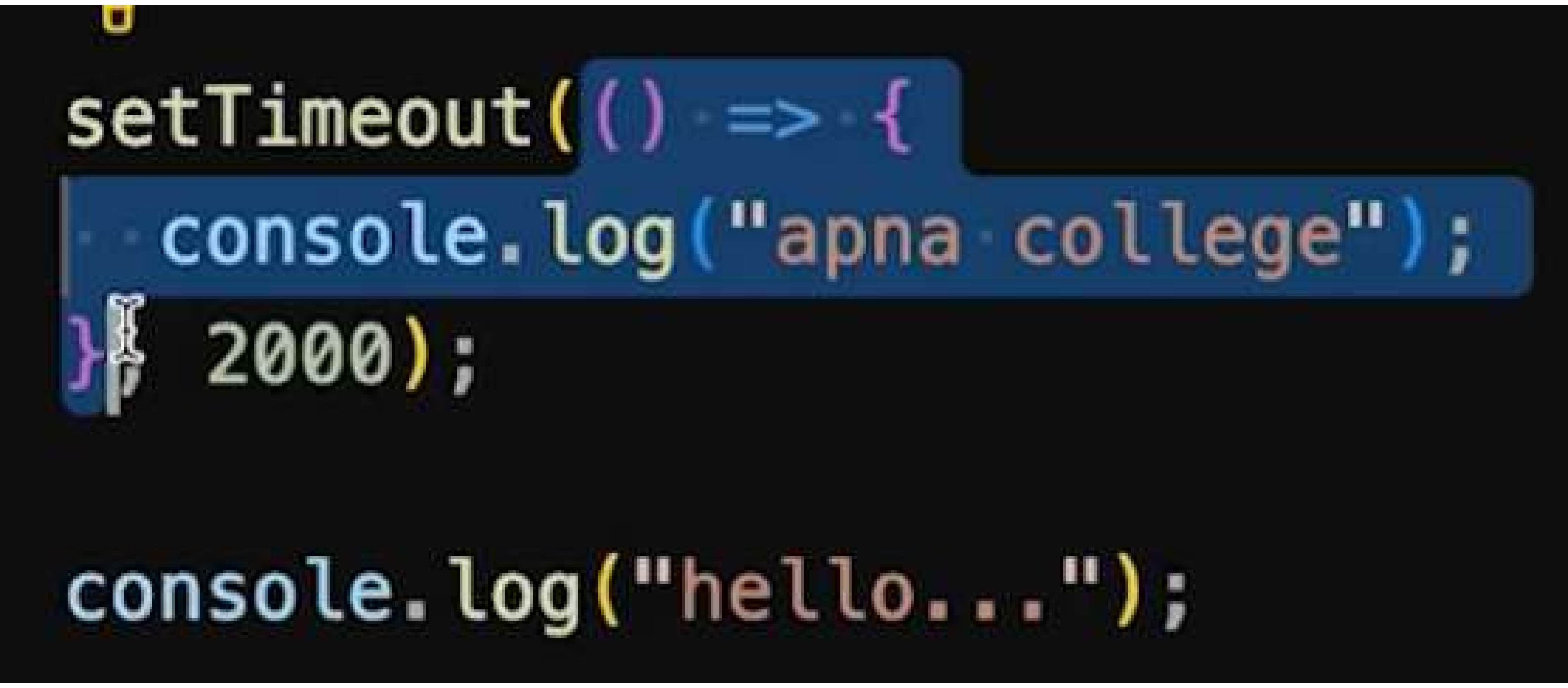
JS app.js > ...

```
let a = 25;
console.log(a);
let b = 10;
console.log(b);
console.log(a + b);
```

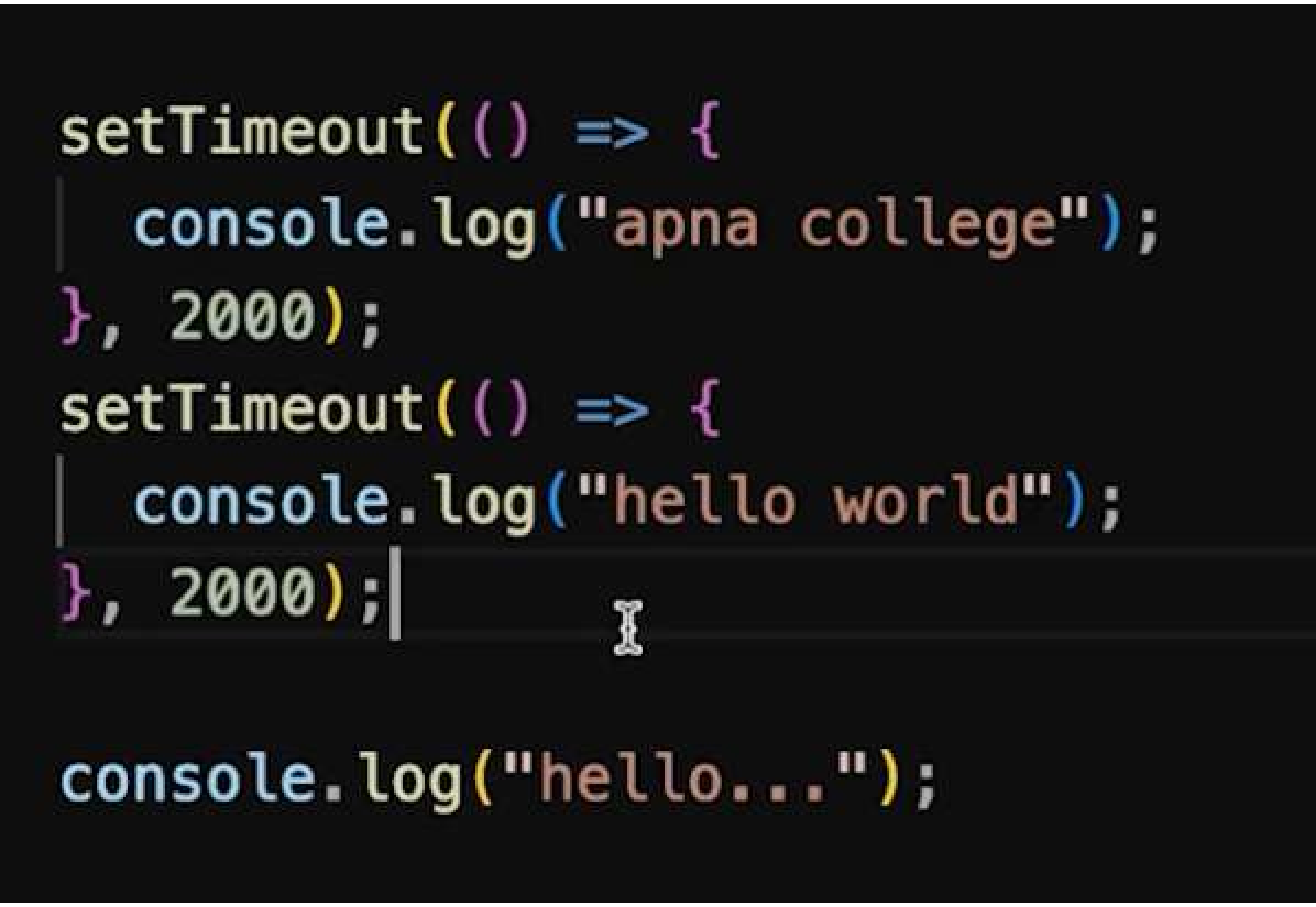


JS is Single Threaded

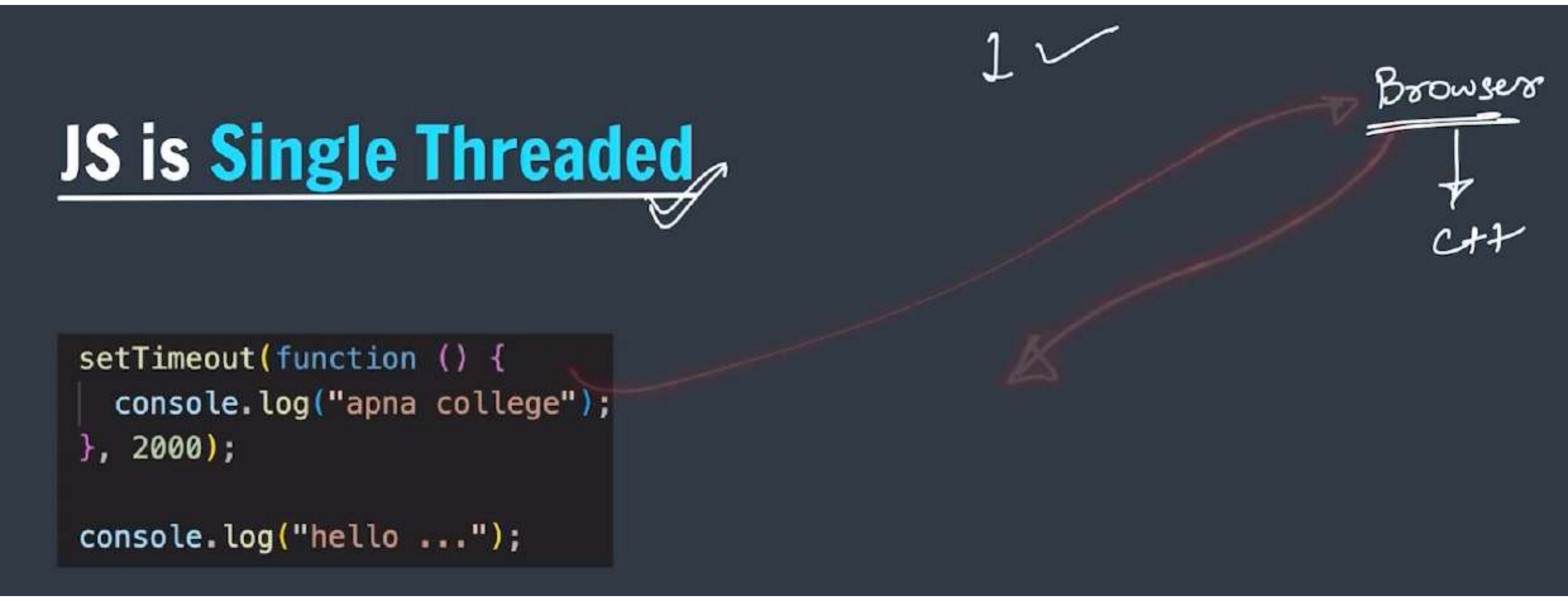
```
setTimeout(function () {
  console.log("apna college");
}, 2000);

console.log("hello ...");
```



```
setTimeout(() => {
  console.log("apna college");
}, 2000);

console.log("hello...");
```

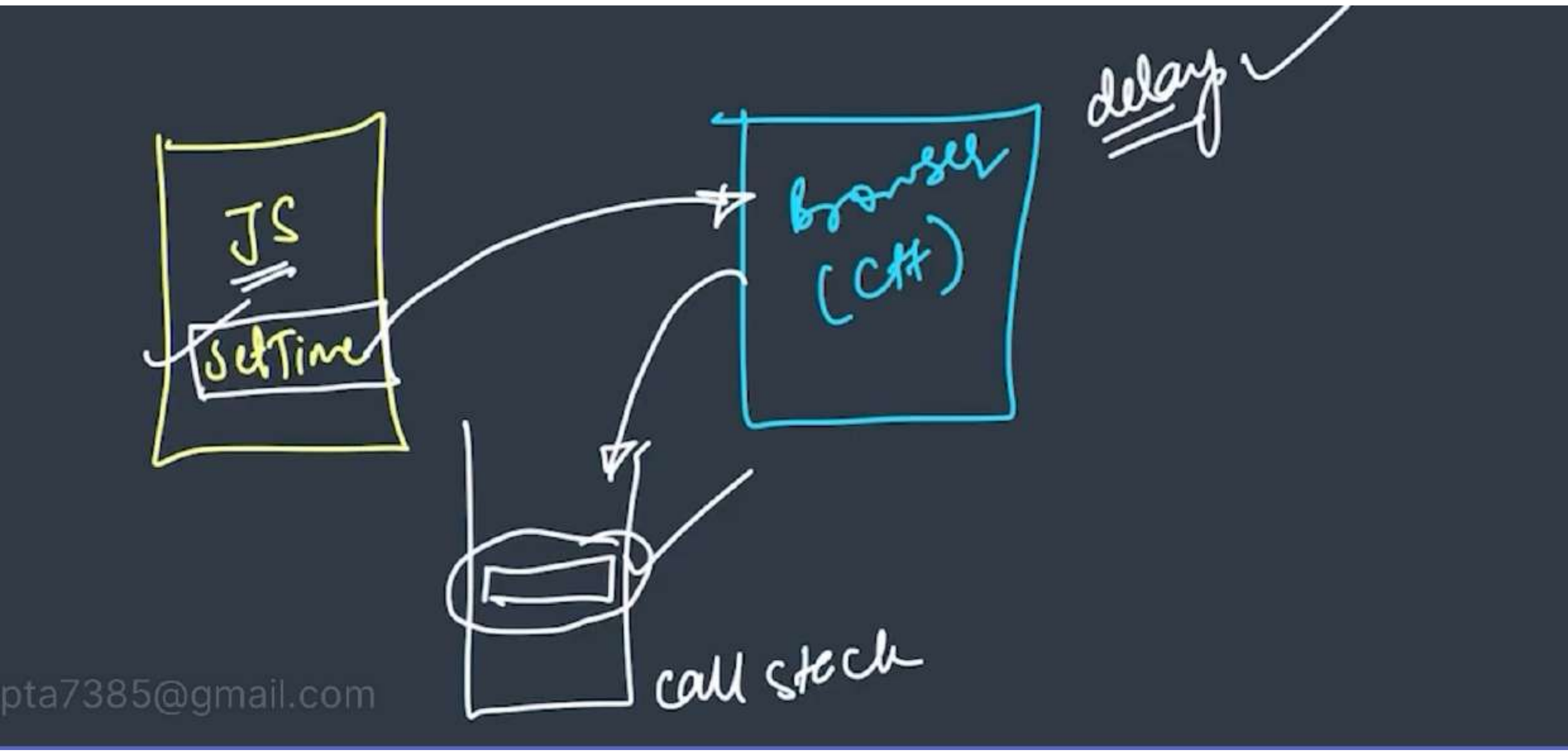


```
setTimeout(() => {
  console.log("apna college");
}, 2000);
setTimeout(() => {
  console.log("hello world");
}, 2000);

console.log("hello...");
```

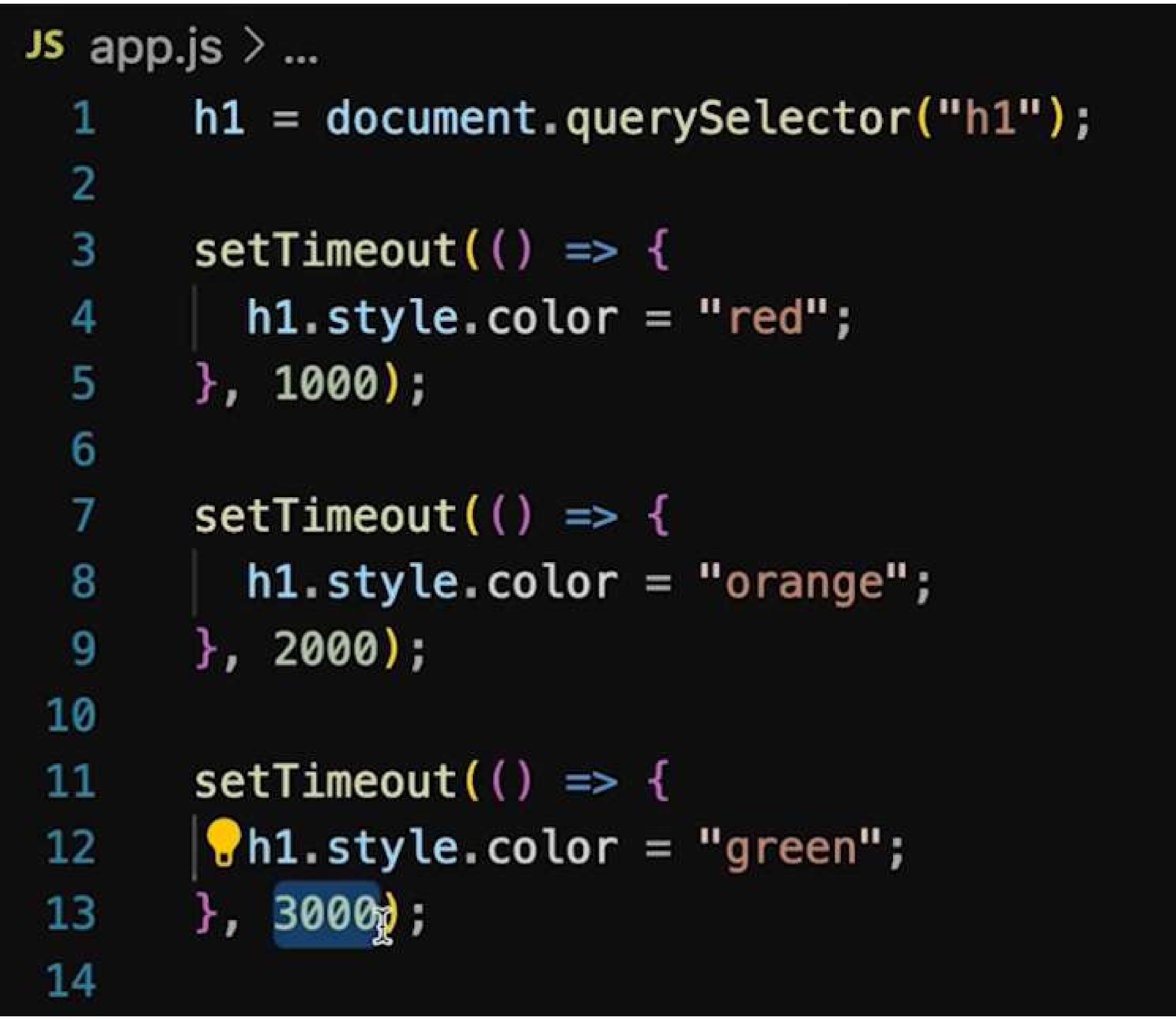


JS is Single Threaded

```
setTimeout(function () {
  console.log("apna college");
}, 2000);

console.log("hello ...");
```

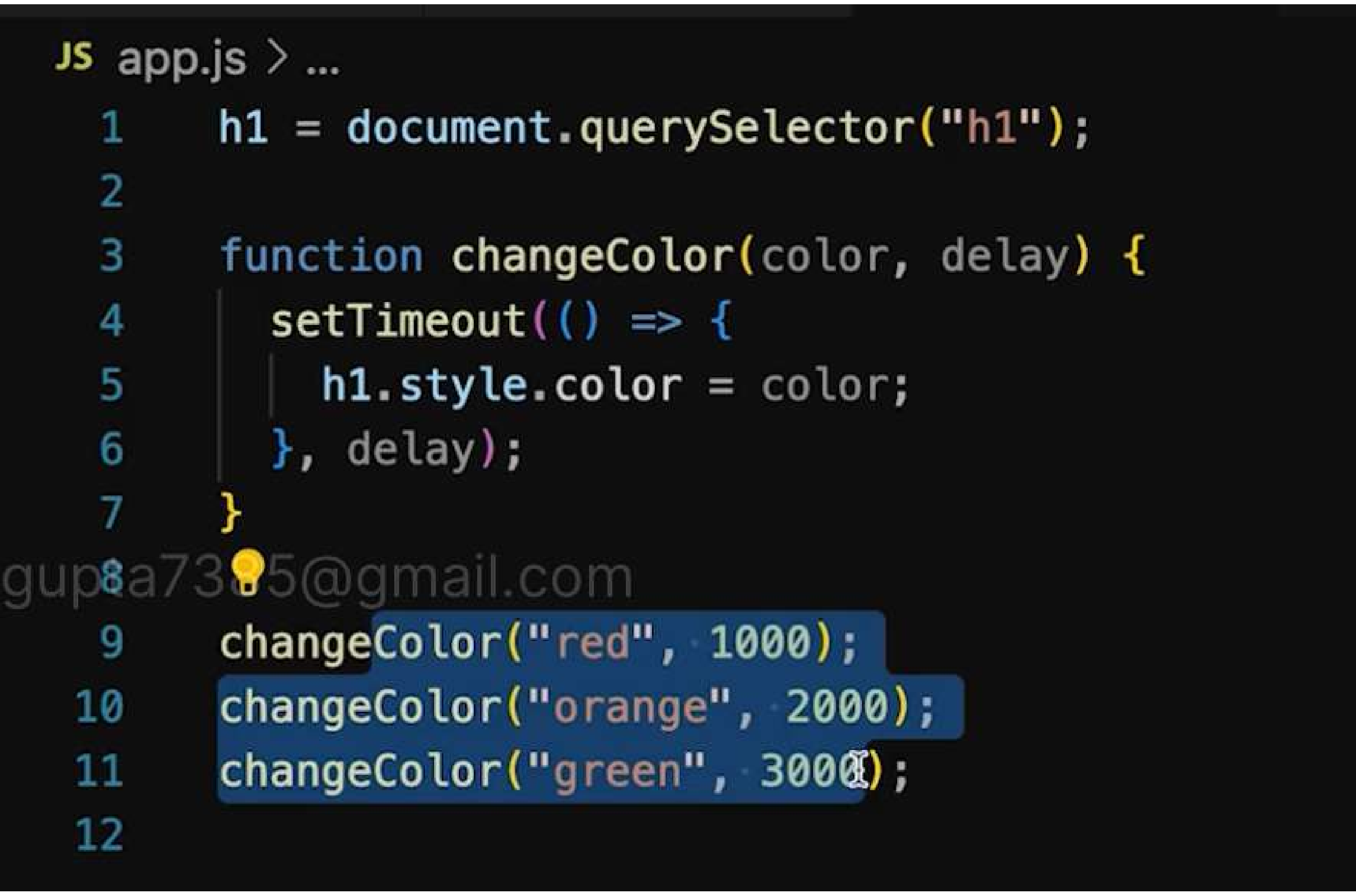

JS
setTime
browser
(C++)
delay
call stack

Callback Hell



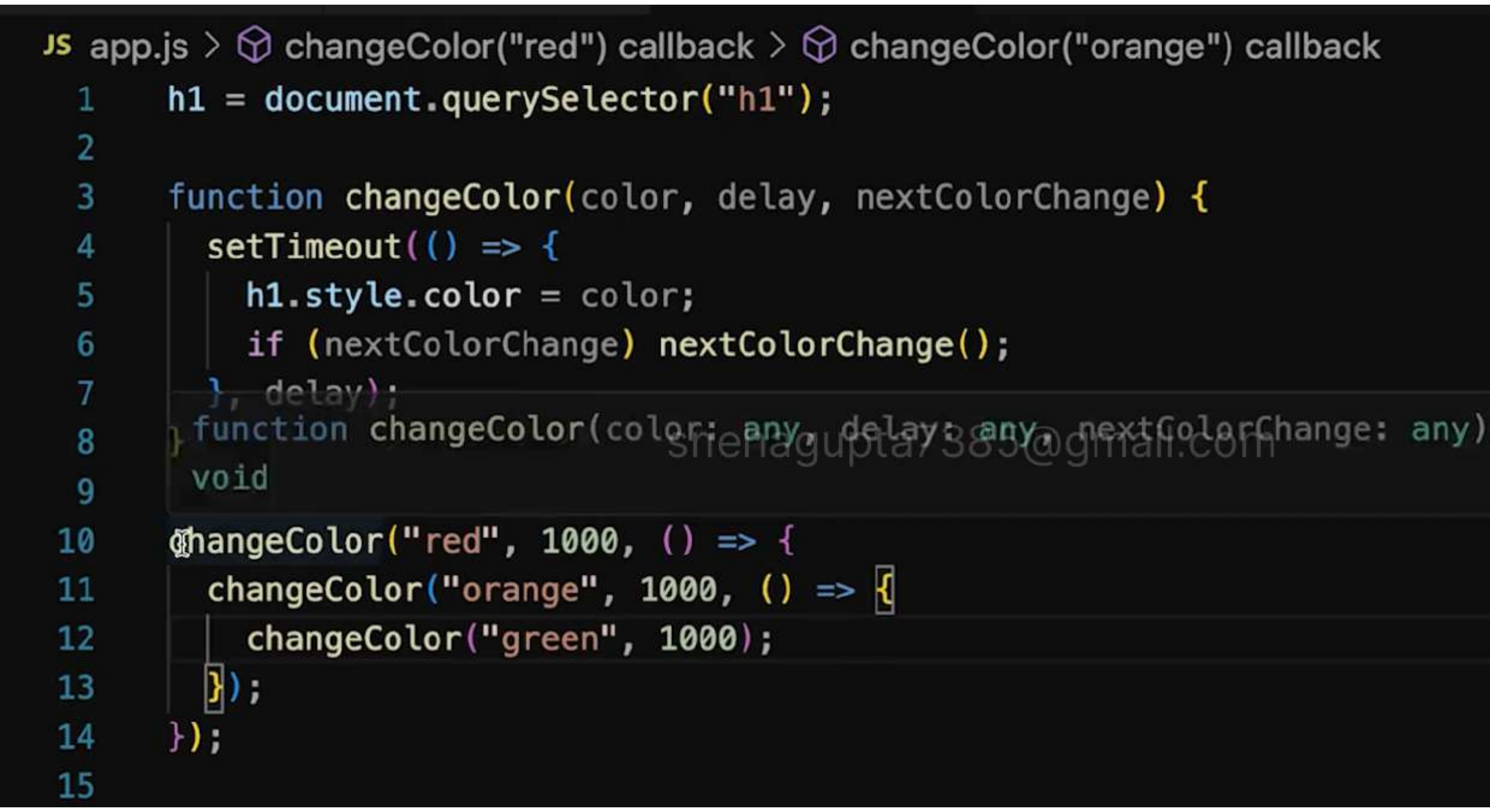
JS app.js > ...

```javascript
h1 = document.querySelector("h1");

setTimeout(() => {
  h1.style.color = "red";
}, 1000);

setTimeout(() => {
  h1.style.color = "orange";
}, 2000);

setTimeout(() => {
  h1.style.color = "green";
}, 3000);

```



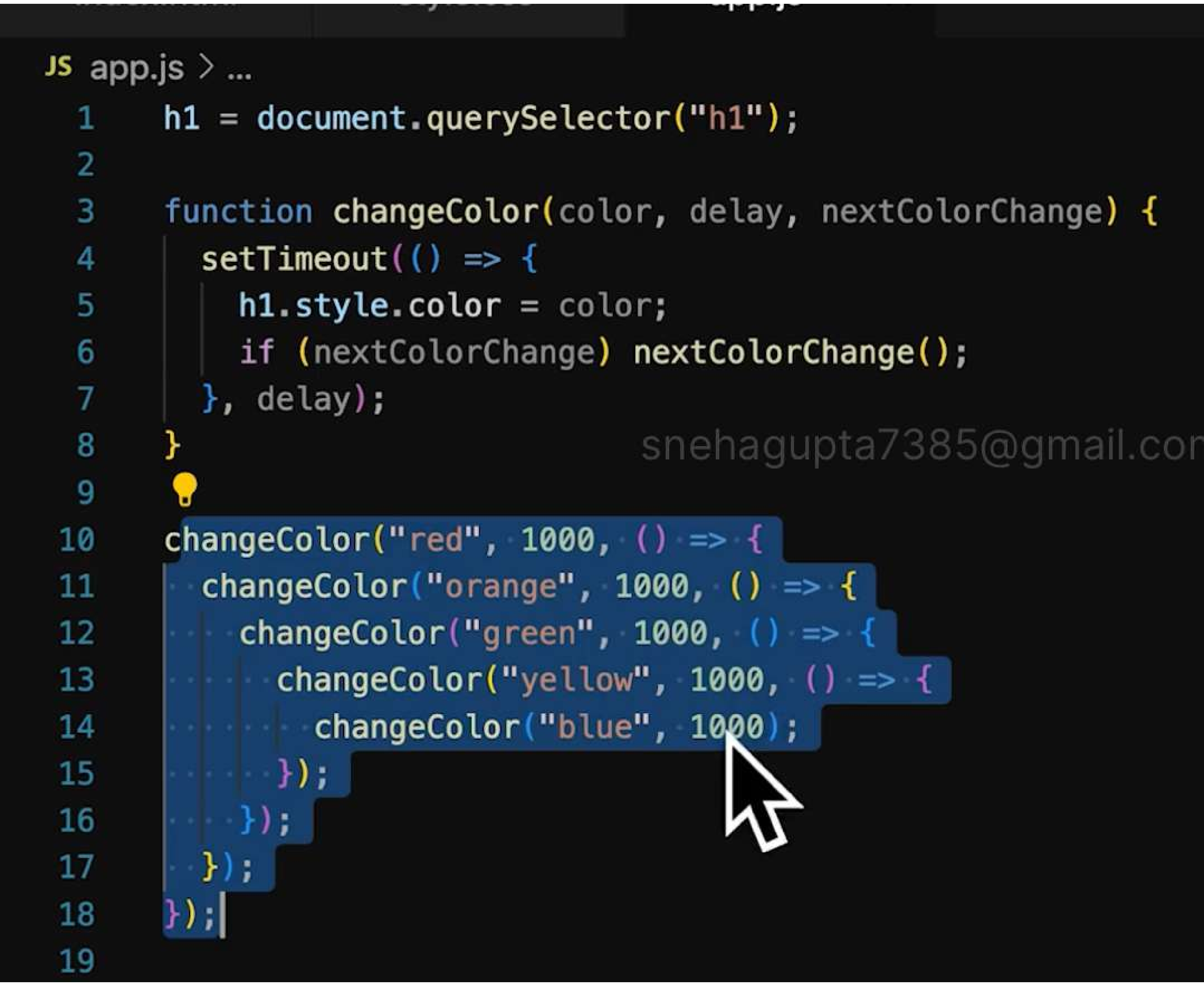
JS app.js > ...

```
h1 = document.querySelector("h1");

function changeColor(color, delay) {
  setTimeout(() => {
    h1.style.color = color;
  }, delay);
}

changeColor("red", 1000);
changeColor("orange", 2000);
changeColor("green", 3000);

```

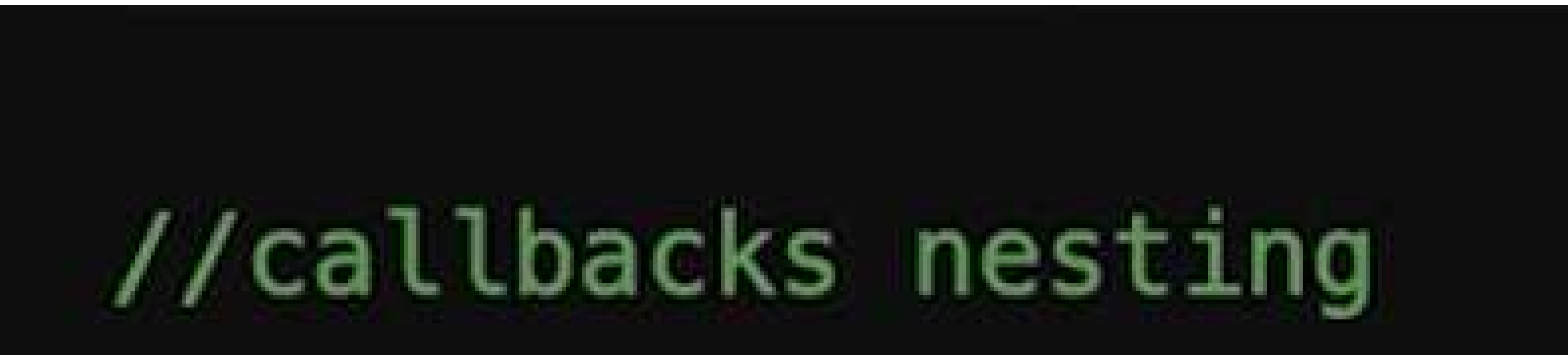

JS app.js > changeColor("red") callback > changeColor("orange") callback

```javascript
h1 = document.querySelector("h1");

function changeColor(color, delay, nextColorChange) {
  setTimeout(() => {
    h1.style.color = color;
    if (nextColorChange) nextColorChange();
  }, delay);
}

changeColor("red", 1000, () => {
  changeColor("orange", 1000, () => {
    changeColor("green", 1000);
  });
});
```

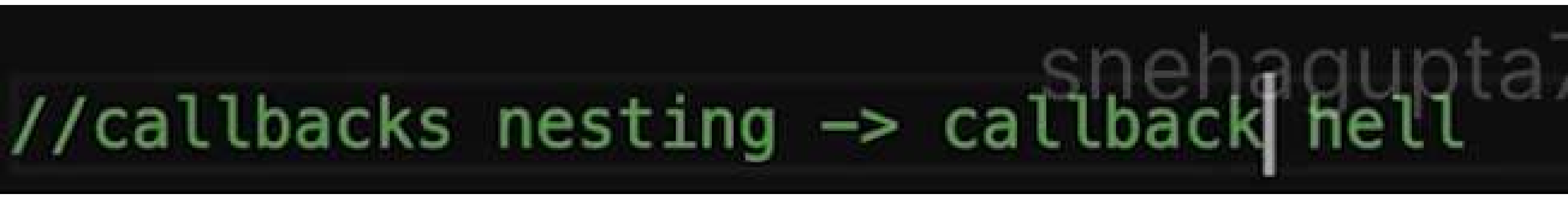
```
function changeColor(color: any, delay: any, nextColorChange: any)
void
```



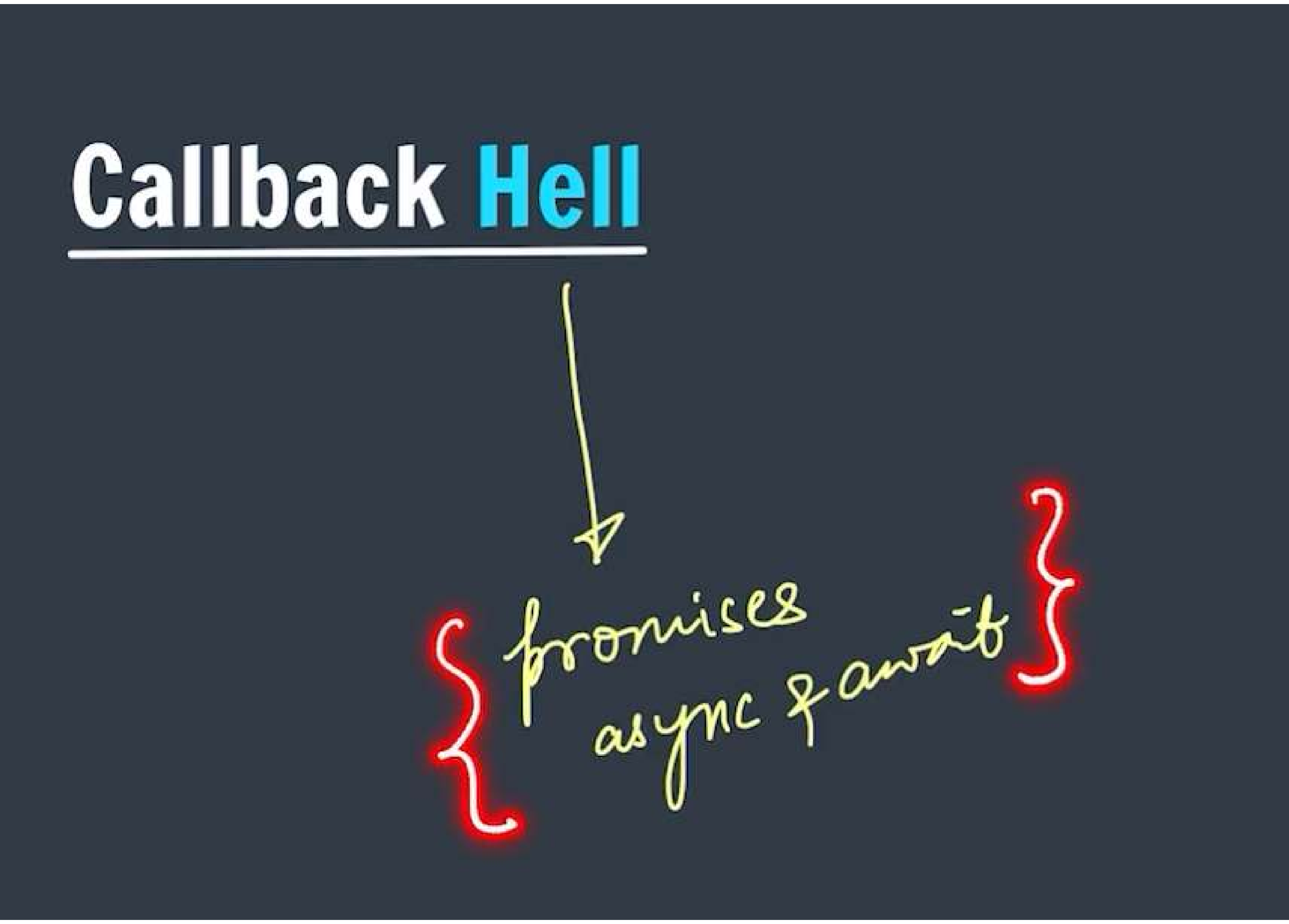
```
h1 = document.querySelector("h1");

function changeColor(color, delay, nextColorChange) {
  setTimeout(() => {
    h1.style.color = color;
    if (nextColorChange) nextColorChange();
  }, delay);
}

changeColor("red", 1000, () => {
  changeColor("orange", 1000, () => {
    changeColor("green", 1000, () => {
      changeColor("yellow", 1000, () => {
        changeColor("blue", 1000);
      });
    });
  });
});
```



```
//callbacks nesting
```

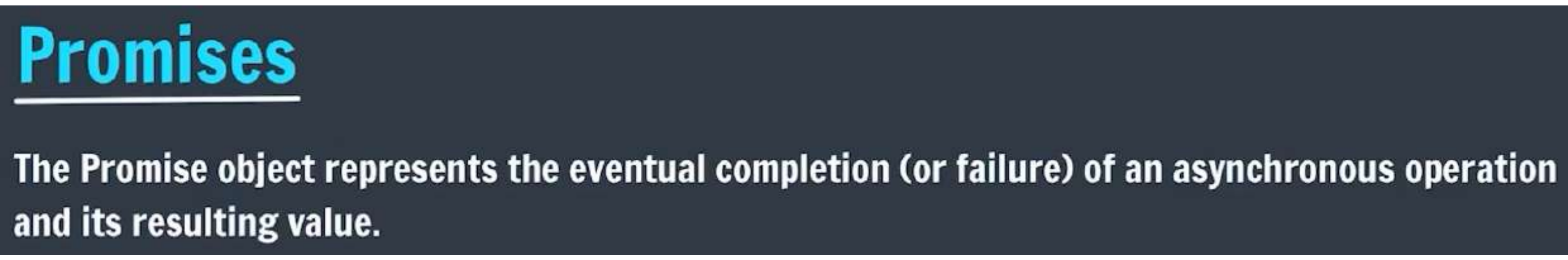


```
//callbacks nesting -> callback hell
```



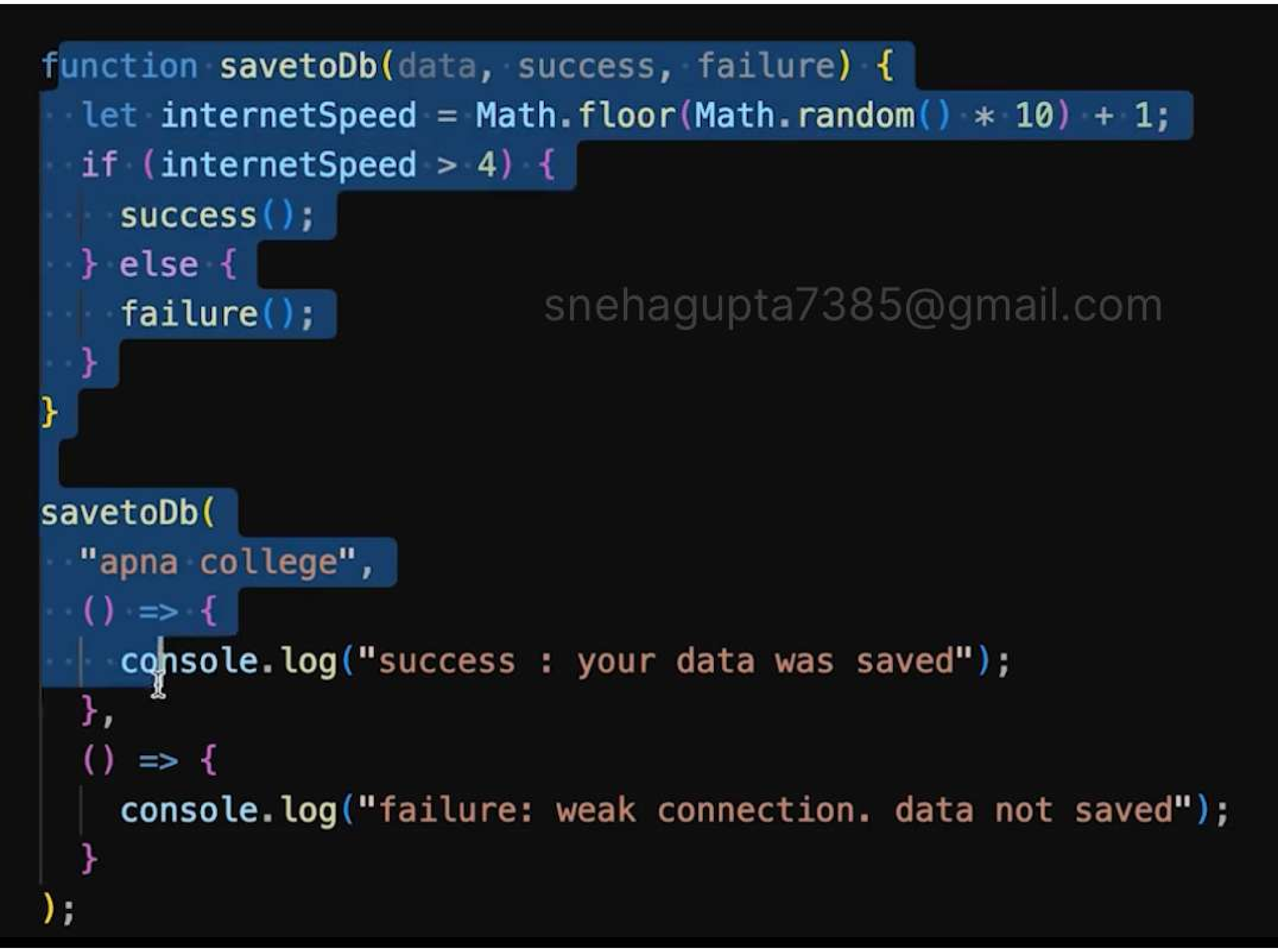
Callback Hell

$$\begin{tikzcd}
\text{Callback Hell} \arrow[d] \\
\left\{ \begin{array}{l} \text{promises} \\ \text{async \& await} \end{array} \right\}
\end{tikzcd}$$

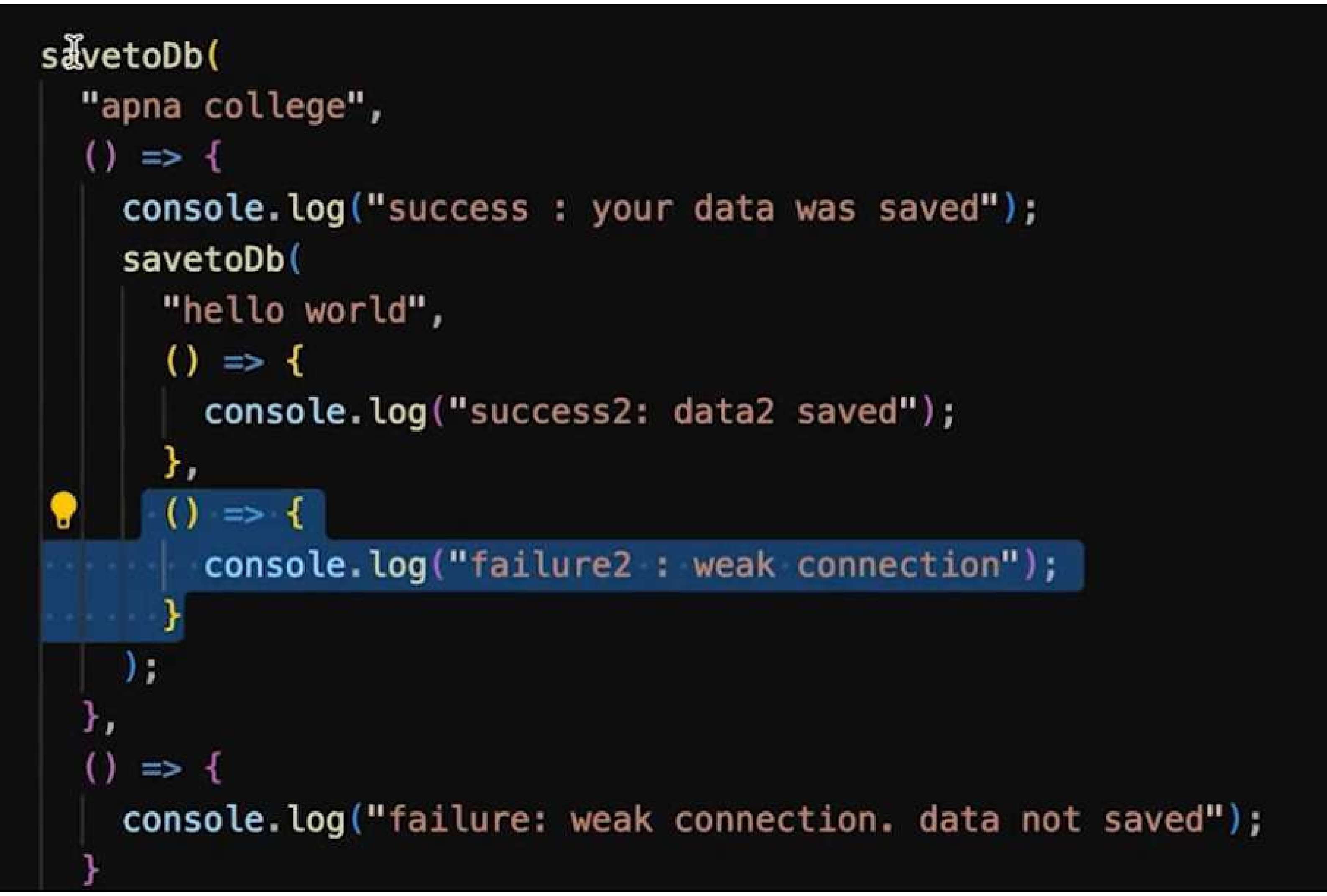


Promises

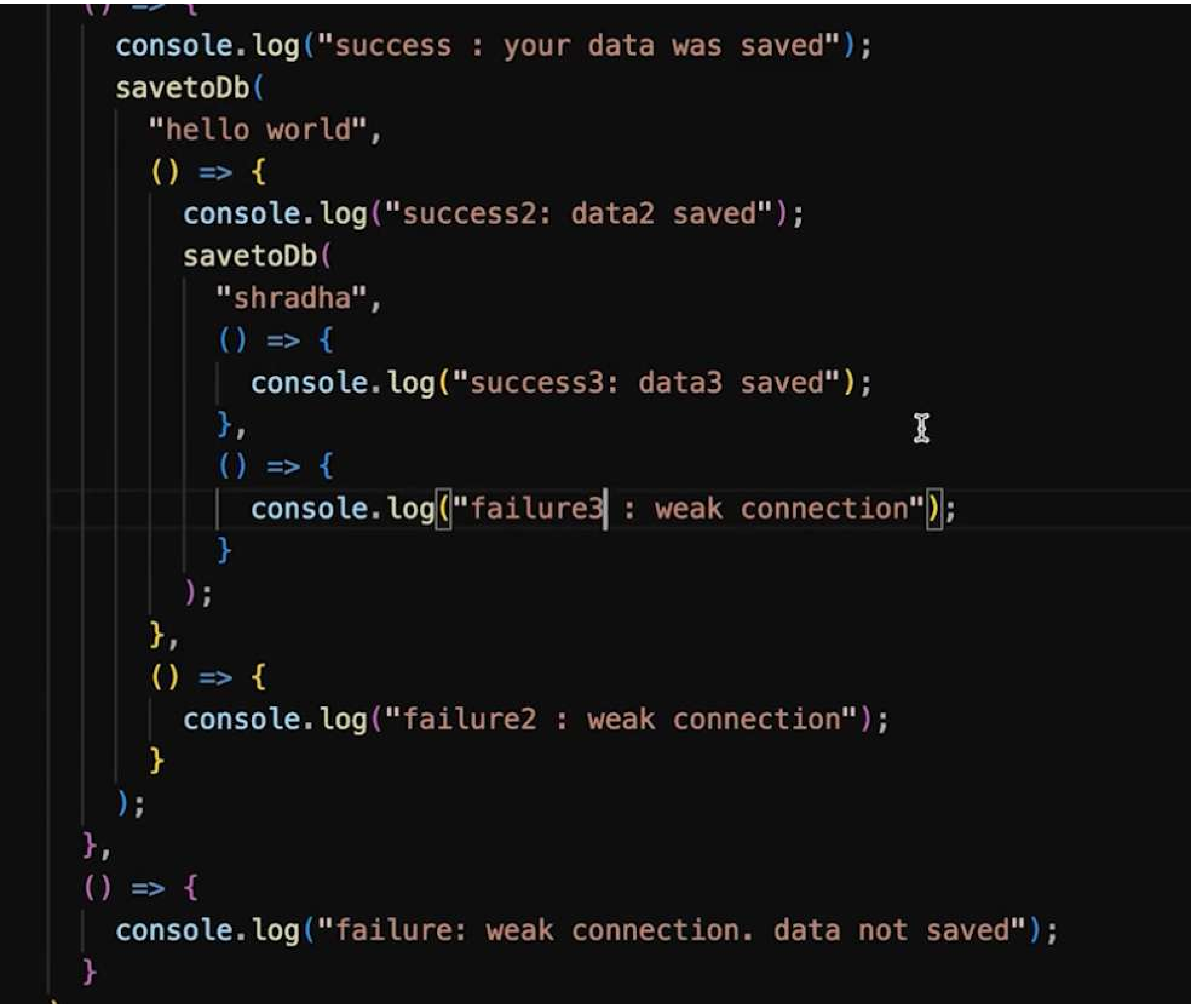
The Promise object represents the eventual completion (or failure) of an asynchronous operation and its resulting value.



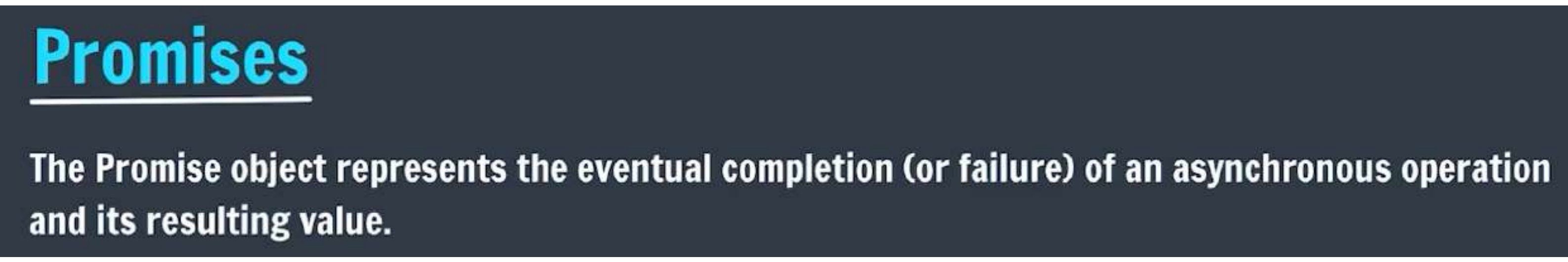
```javascript
function savetoDb(data, success, failure) {
  let internetSpeed = Math.floor(Math.random() * 10) + 1;
  if (internetSpeed > 4) {
    success();
  } else {
    failure();
  }
}

savetoDb(
  "apna college",
  () => {
    console.log("success : your data was saved");
  },
  () => {
    console.log("failure: weak connection. data not saved");
  }
);
```



```
savetoDb(
  "apna college",
  () => {
    console.log("success : your data was saved");
    savetoDb(
      "hello world",
      () => {
        console.log("success2: data2 saved");
      },
      () => {
        console.log("failure2 : weak connection");
      }
    );
  },
  () => {
    console.log("failure: weak connection. data not saved");
  }
```

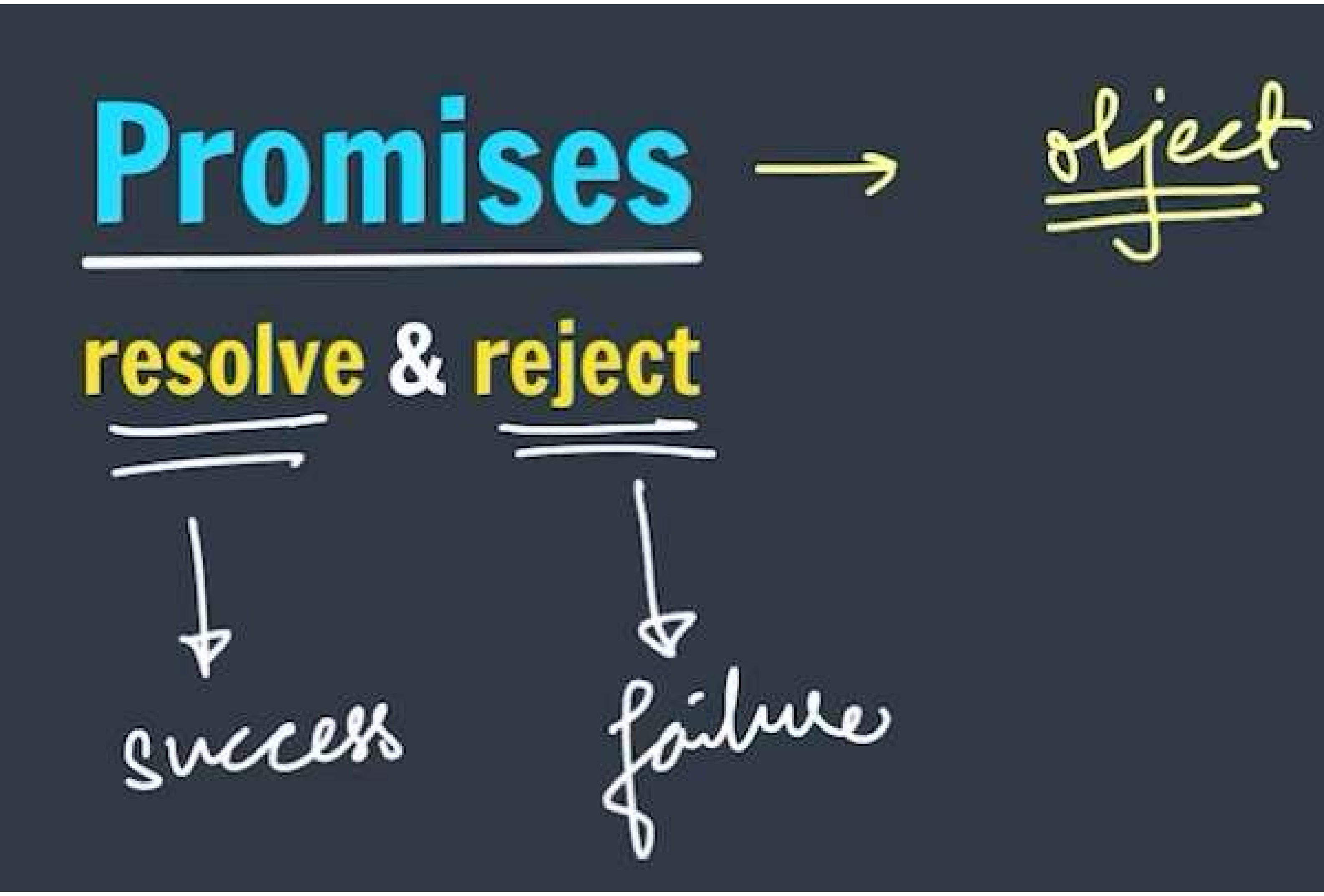


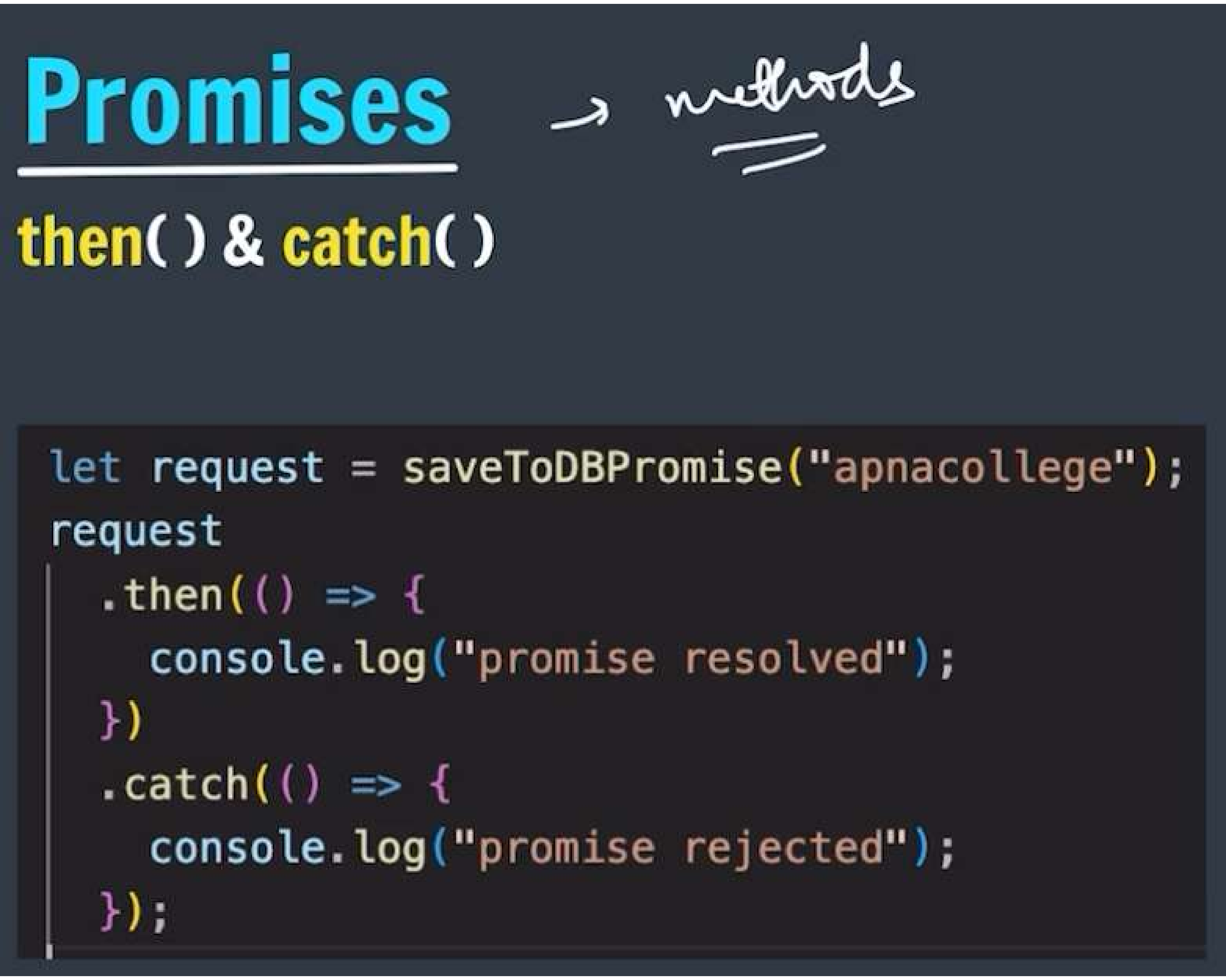
```
  () => {
    console.log("success : your data was saved");
    savetoDb(
      "hello world",
      () => {
        console.log("success2: data2 saved");
        savetoDb(
          "shradha",
          () => {
            console.log("success3: data3 saved");
          },
          () => {
            console.log("failure3 : weak connection");
          }
        );
      },
      () => {
        console.log("failure2 : weak connection");
      }
    );
  },
  () => {
    console.log("failure: weak connection. data not saved");
  }
```



Promises

The Promise object represents the eventual completion (or failure) of an asynchronous operation and its resulting value.

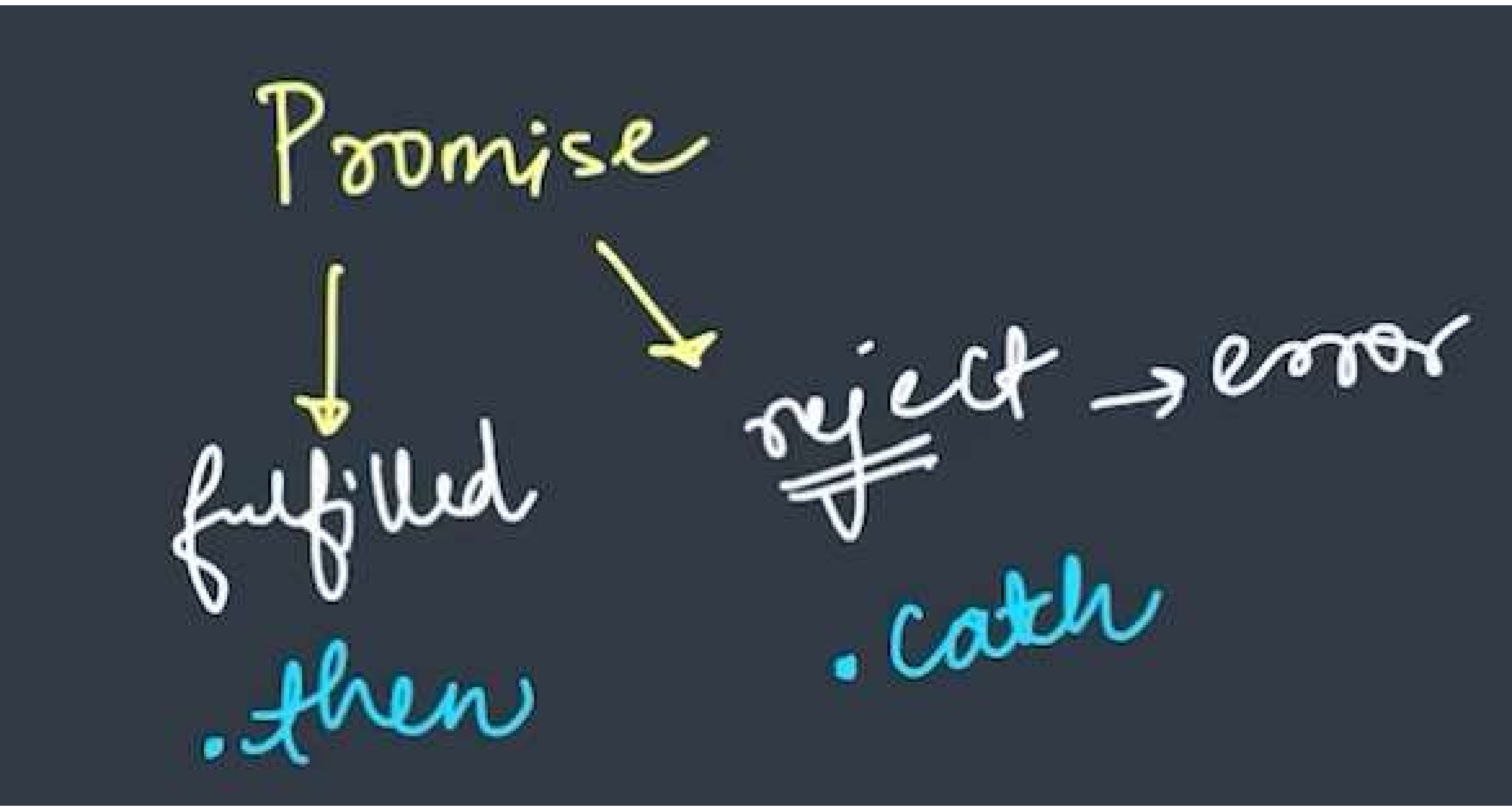

Promises
object
resolve & reject
success
failure

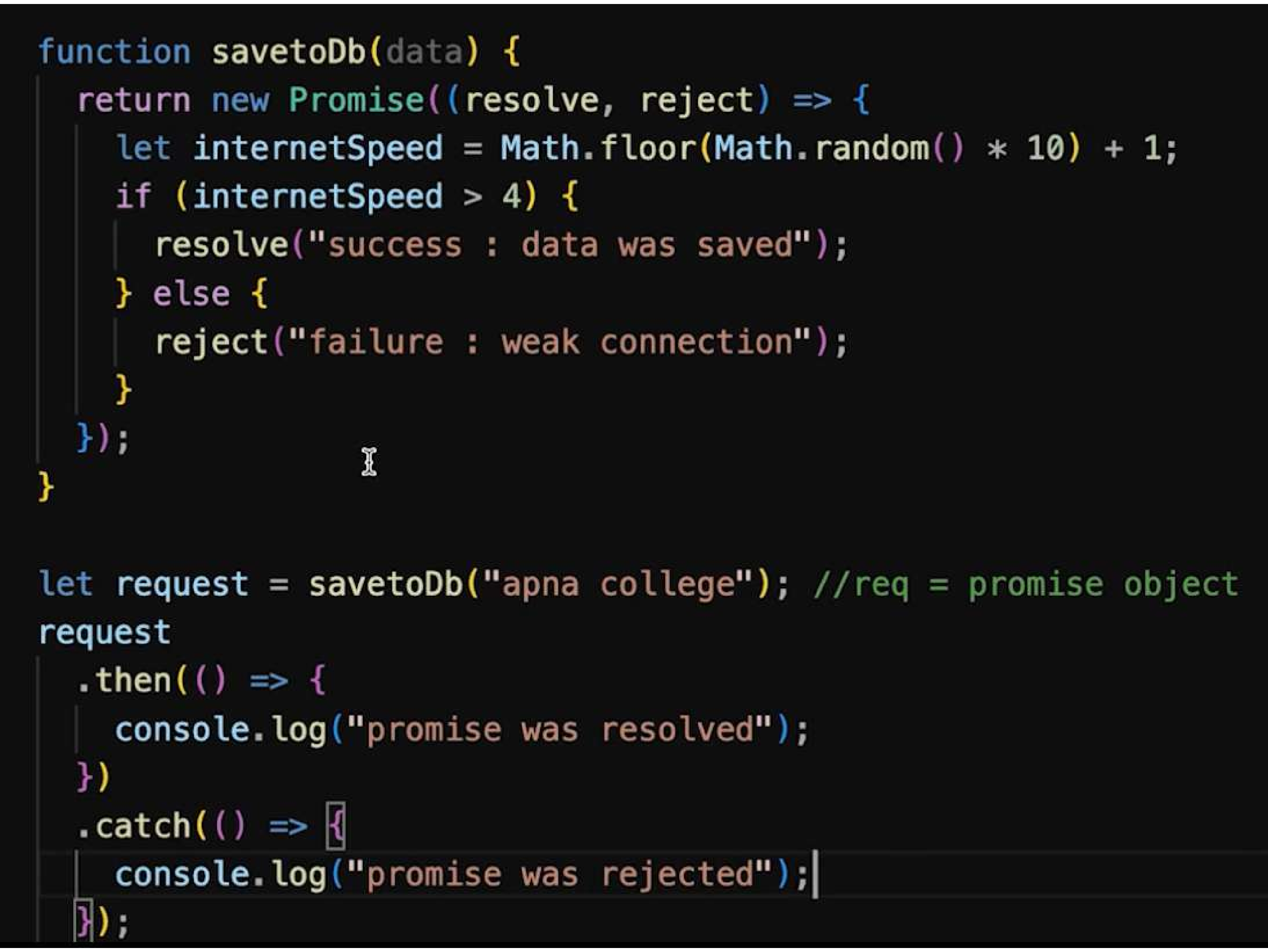


Promises → methods

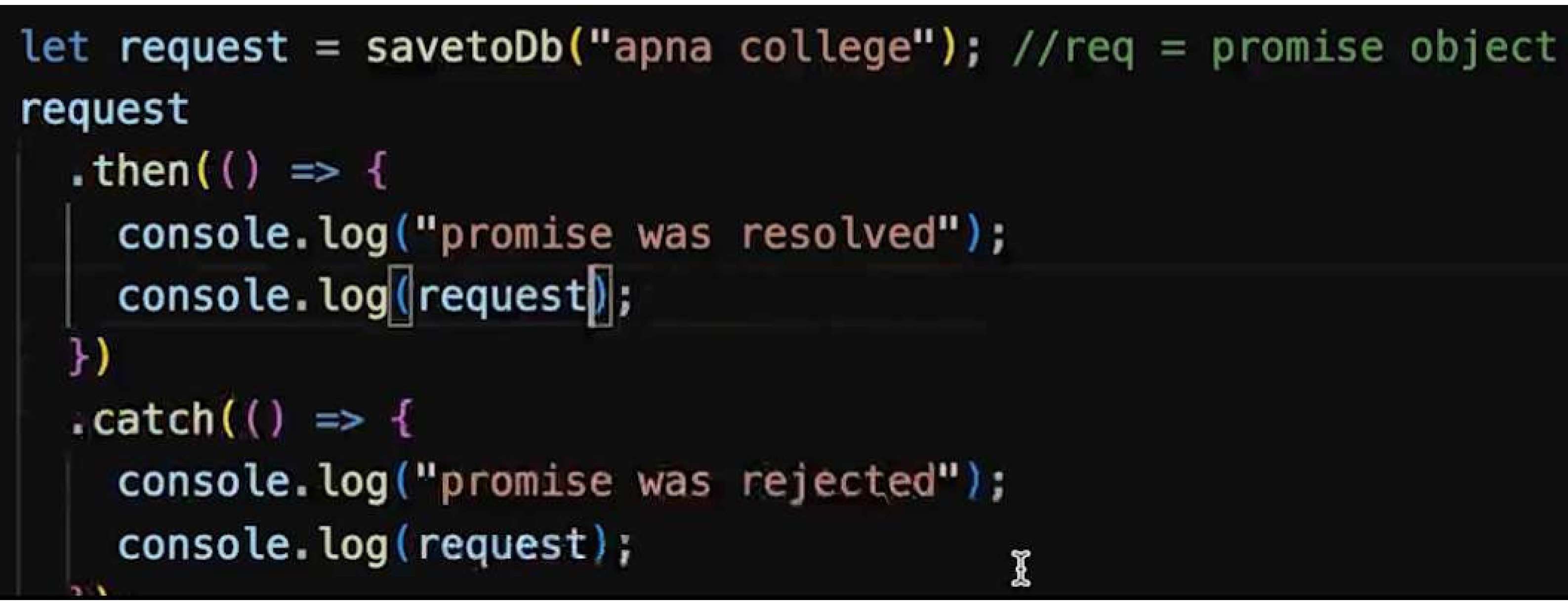
then( ) & catch( )

```
let request = saveToDBPromise("apnacollege");
request
  .then(() => {
    console.log("promise resolved");
  })
  .catch(() => {
    console.log("promise rejected");
  });
```

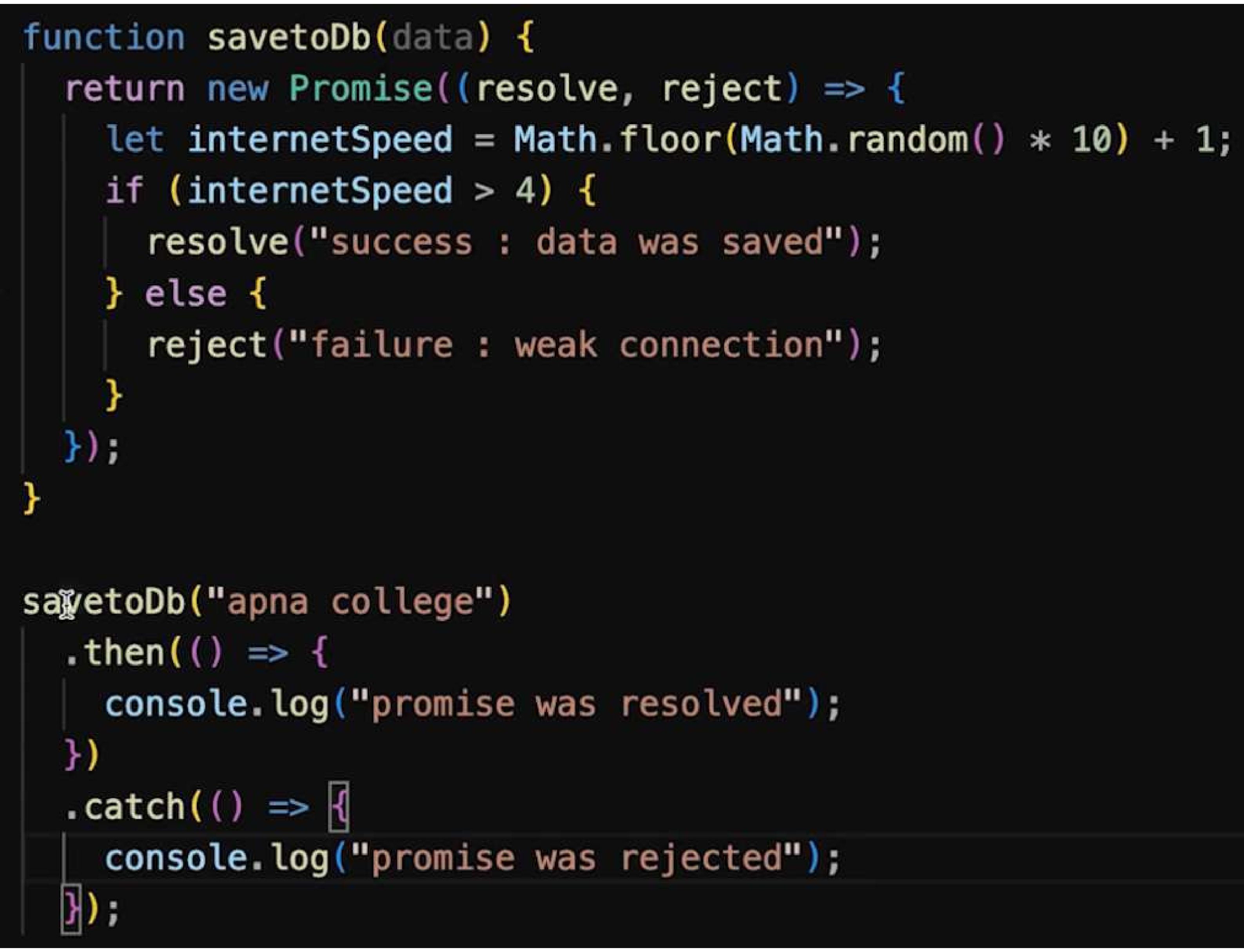

Promise
fulfilled
.then
reject → error
.catch



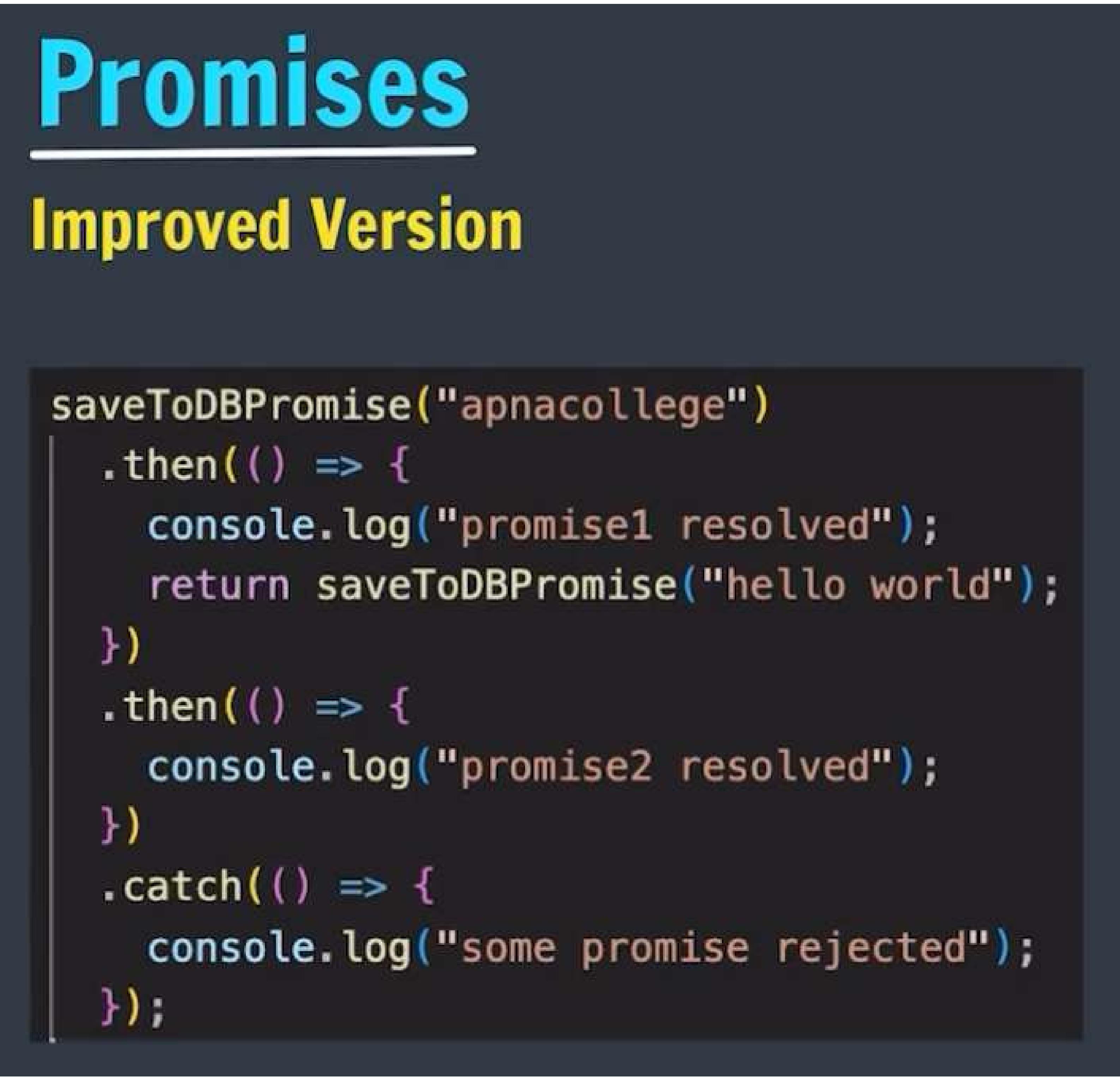
```
function savetoDb(data) {
  return new Promise((resolve, reject) => {
    let internetSpeed = Math.floor(Math.random() * 10) + 1;
    if (internetSpeed > 4) {
      resolve("success : data was saved");
    } else {
      reject("failure : weak connection");
    }
  });
}

let request = savetoDb("apna college"); //req = promise object
request
  .then(() => {
    console.log("promise was resolved");
  })
  .catch(() => {
    console.log("promise was rejected");
  });
```



```
let request = savetoDb("apna college"); //req = promise object
request
  .then(() => {
    console.log("promise was resolved");
    console.log(request);
  })
  .catch(() => {
    console.log("promise was rejected");
    console.log(request);
```



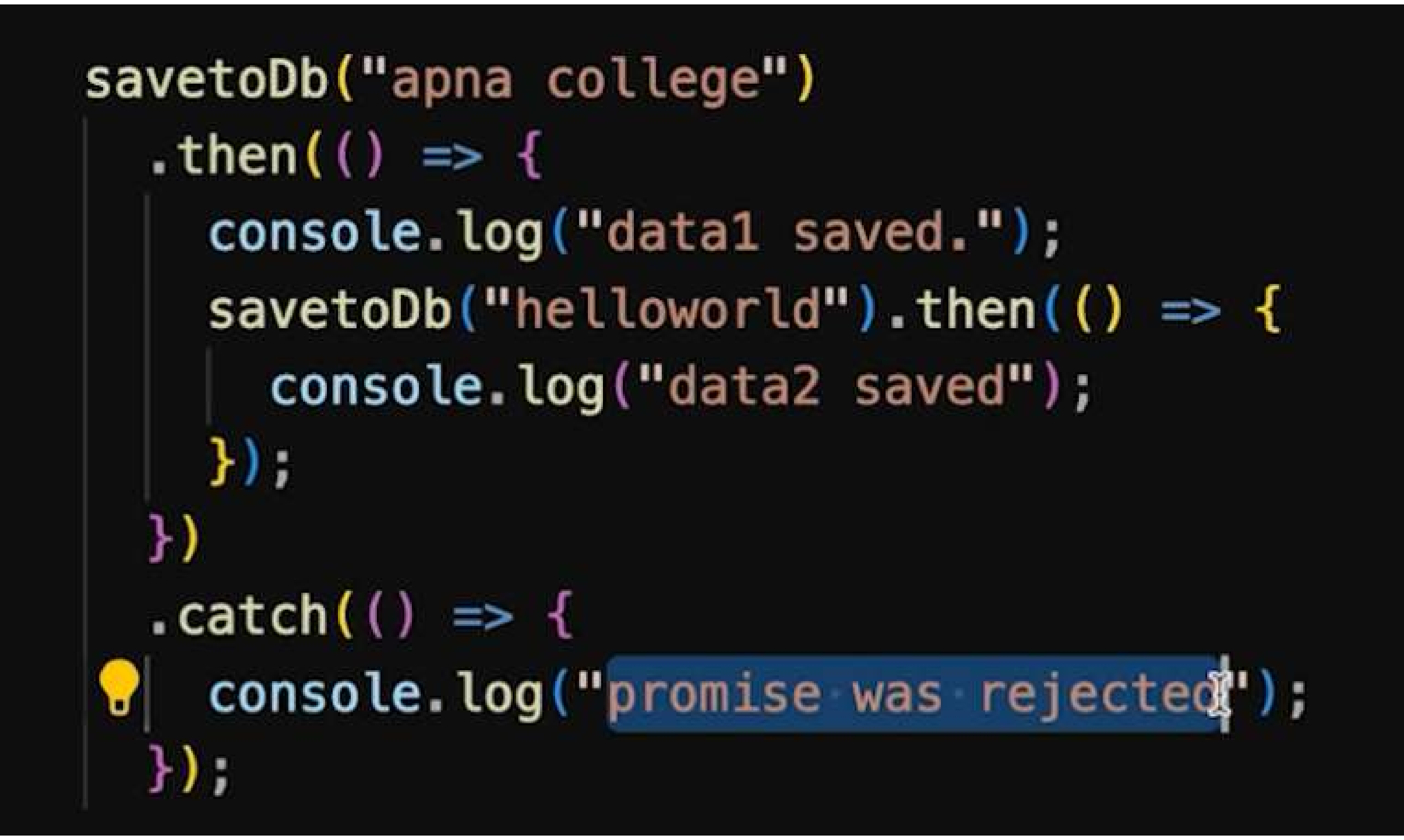
```javascript
function savetoDb(data) {
  return new Promise((resolve, reject) => {
    let internetSpeed = Math.floor(Math.random() * 10) + 1;
    if (internetSpeed > 4) {
      resolve("success : data was saved");
    } else {
      reject("failure : weak connection");
    }
  });
}

savetoDb("apna college")
  .then(() => {
    console.log("promise was resolved");
  })
  .catch(() => {
    console.log("promise was rejected");
  });
```



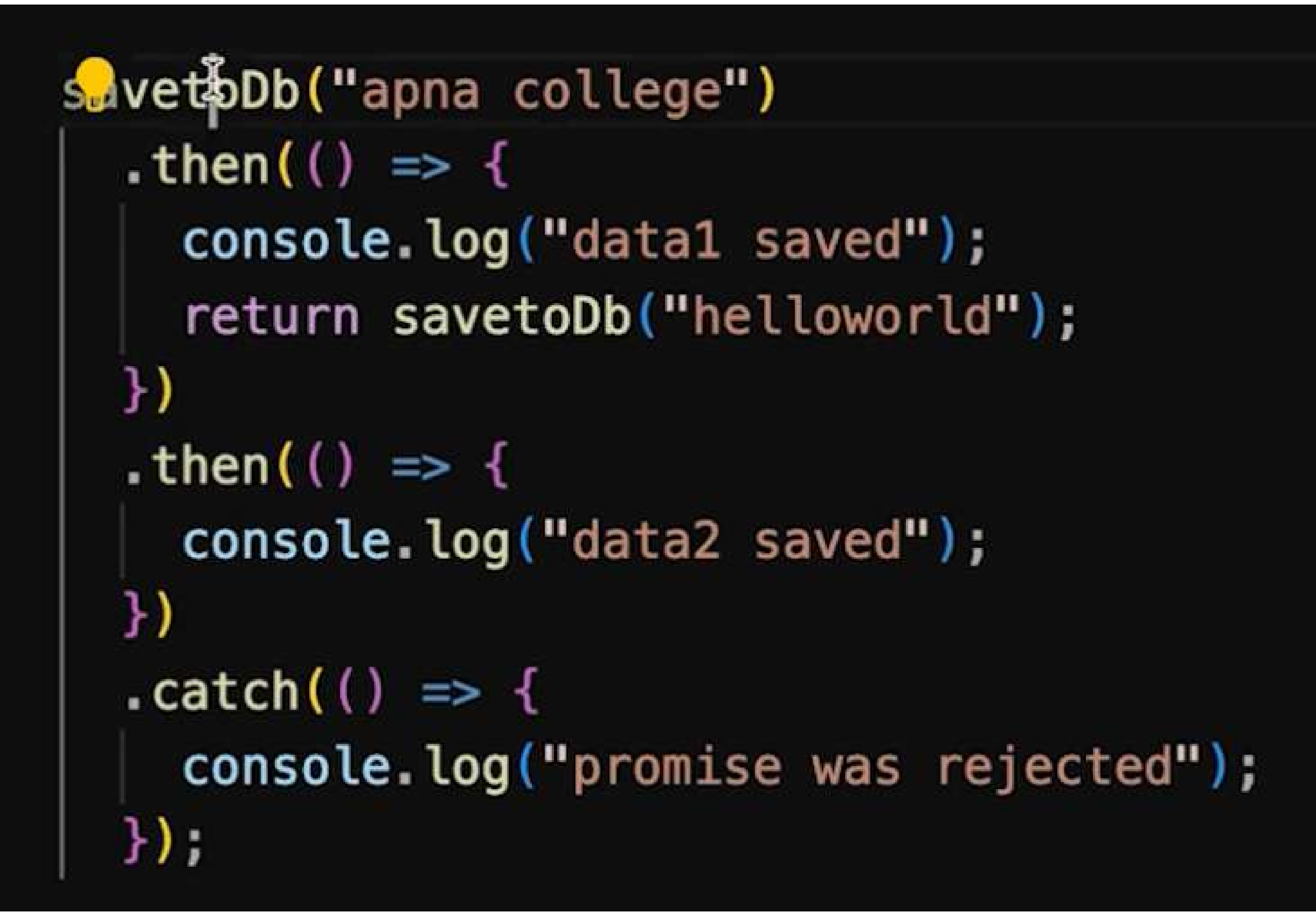
Promises

Improved Version

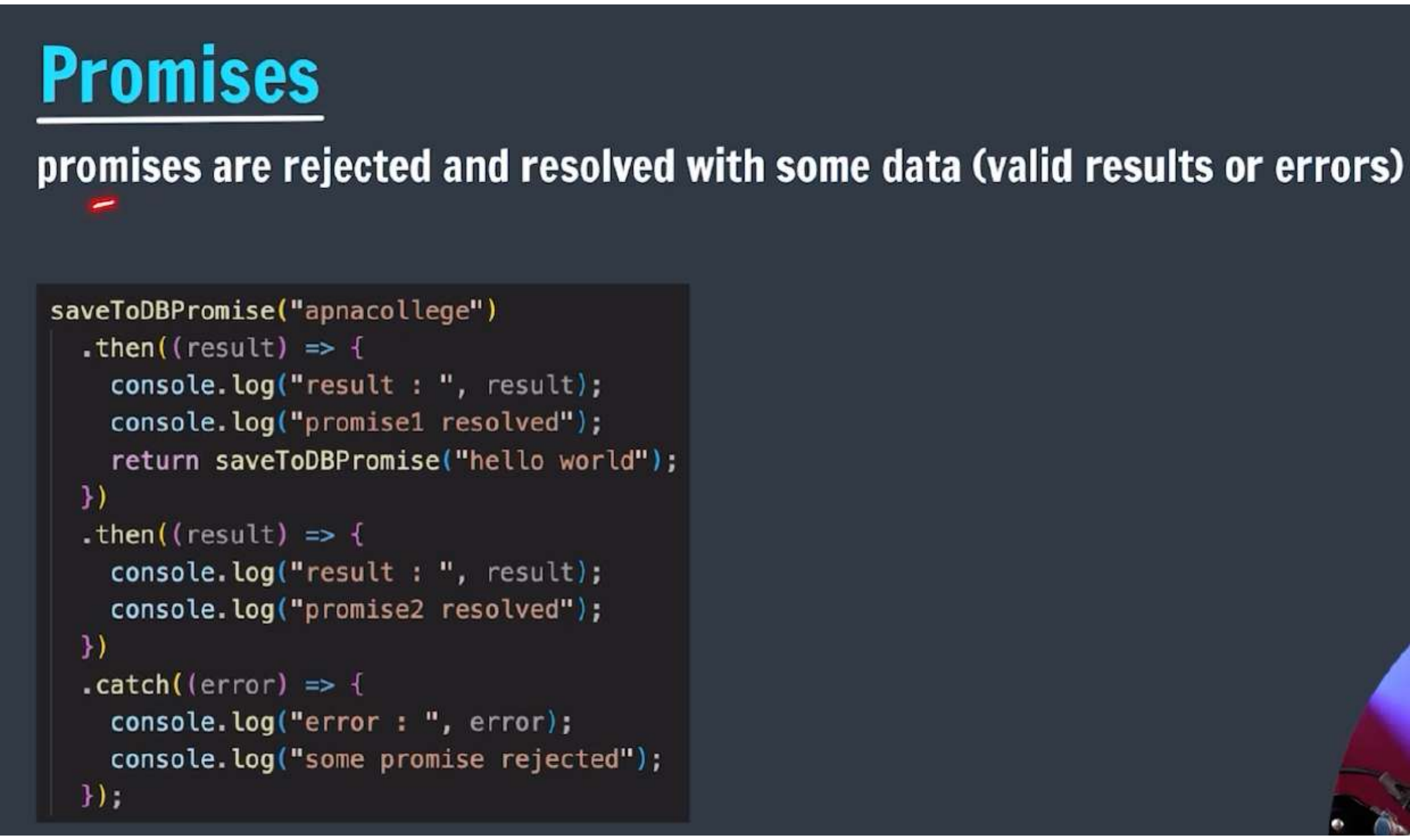
```
saveToDBPromise("apnacollege")
  .then(() => {
    console.log("promise1 resolved");
    return saveToDBPromise("hello world");
  })
  .then(() => {
    console.log("promise2 resolved");
  })
  .catch(() => {
    console.log("some promise rejected");
  });
```



```
savetoDb("apna college")
  .then(() => {
    console.log("data1 saved.");
    savetoDb("helloworld").then(() => {
      console.log("data2 saved");
    });
  })
  .catch(() => {
    console.log("promise was rejected");
  });
```



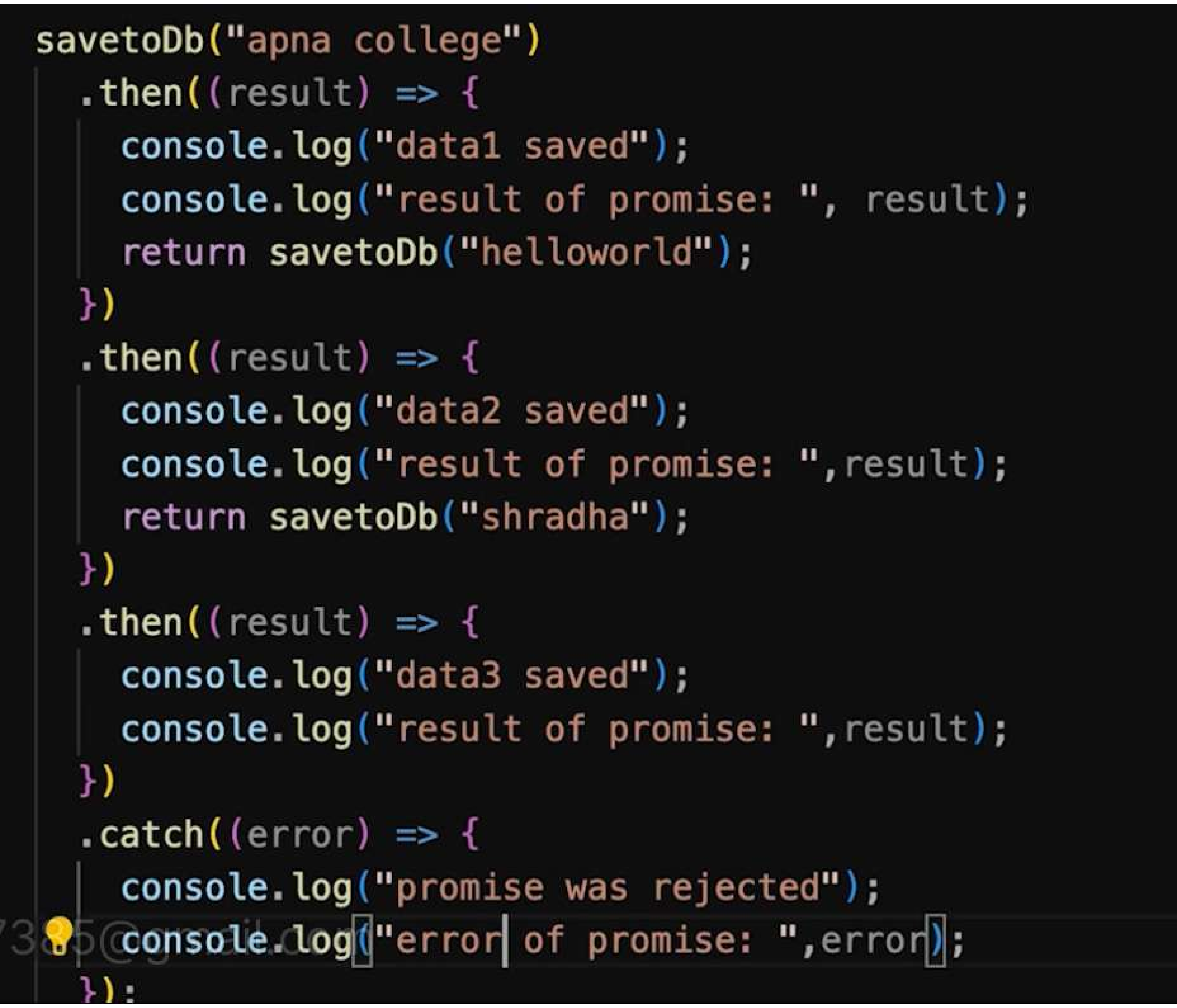
```
savetoDb("apna college")
  .then(() => {
    console.log("data1 saved");
    return savetoDb("helloworld");
  })
  .then(() => {
    console.log("data2 saved");
  })
  .catch(() => {
    console.log("promise was rejected");
  });
```



Promises

promises are rejected and resolved with some data (valid results or errors)

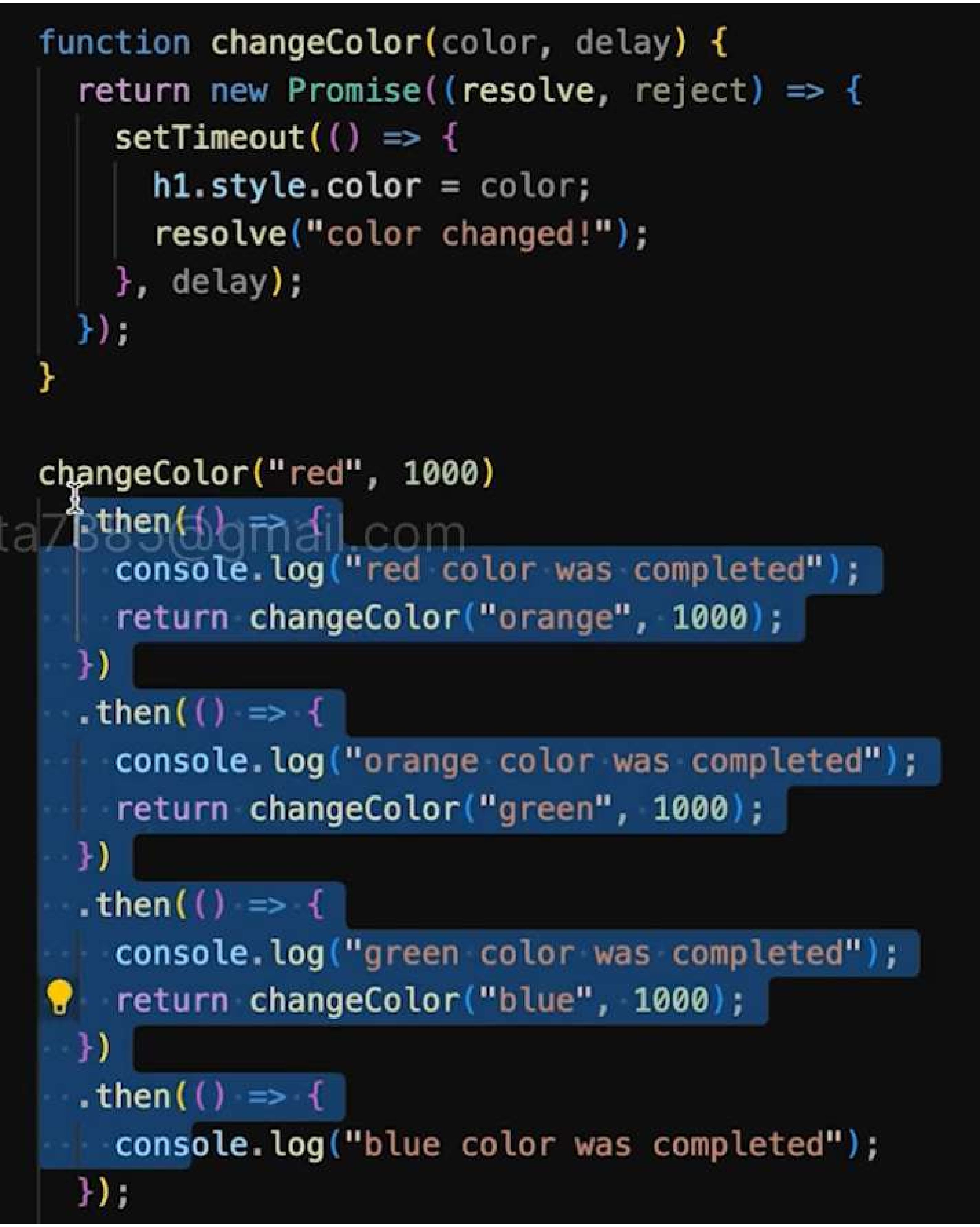
```
saveToDBPromise("apnacollege")
  .then((result) => {
    console.log("result : ", result);
    console.log("promise1 resolved");
    return saveToDBPromise("hello world");
  })
  .then((result) => {
    console.log("result : ", result);
    console.log("promise2 resolved");
  })
  .catch((error) => {
    console.log("error : ", error);
    console.log("some promise rejected");
  });
```



```
savetoDb("apna college")
  .then((result) => {
    console.log("data1 saved");
    console.log("result of promise: ", result);
    return savetoDb("helloworld");
  })
  .then((result) => {
    console.log("data2 saved");
    console.log("result of promise: ",result);
    return savetoDb("shradha");
  })
  .then((result) => {
    console.log("data3 saved");
    console.log("result of promise: ",result);
  })
  .catch((error) => {
    console.log("promise was rejected");
    console.log("error of promise: ",error);
  });
```

Promises

let's apply promises to our callback hell



```
function changeColor(color, delay) {
  return new Promise((resolve, reject) => {
    setTimeout(() => {
      h1.style.color = color;
      resolve("color changed!");
    }, delay);
  });
}

changeColor("red", 1000)
  .then(() => {
    console.log("red color was completed");
    return changeColor("orange", 1000);
  })
  .then(() => {
    console.log("orange color was completed");
    return changeColor("green", 1000);
  })
  .then(() => {
    console.log("green color was completed");
    return changeColor("blue", 1000);
  })
  .then(() => {
    console.log("blue color was completed");
  });
```