the RISE of async... JavaScript

Jeremy Fairbank

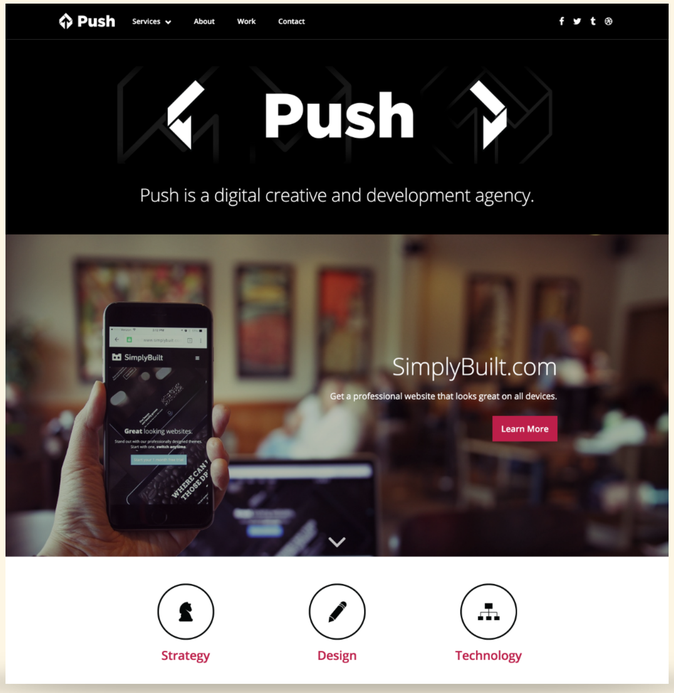
blog.jeremyfairbank.com

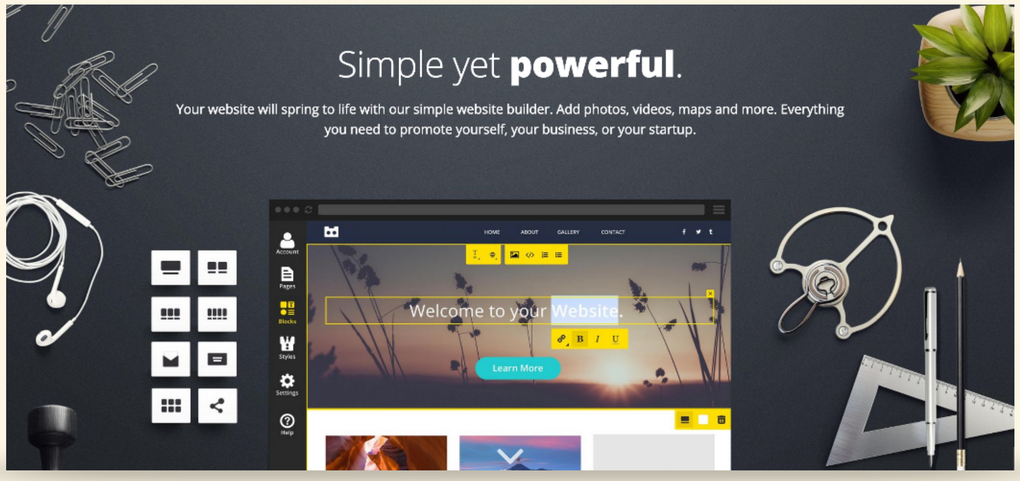
@elpapapollo | gh/jfairbank

Push

We help brands excel.

pushagency.io



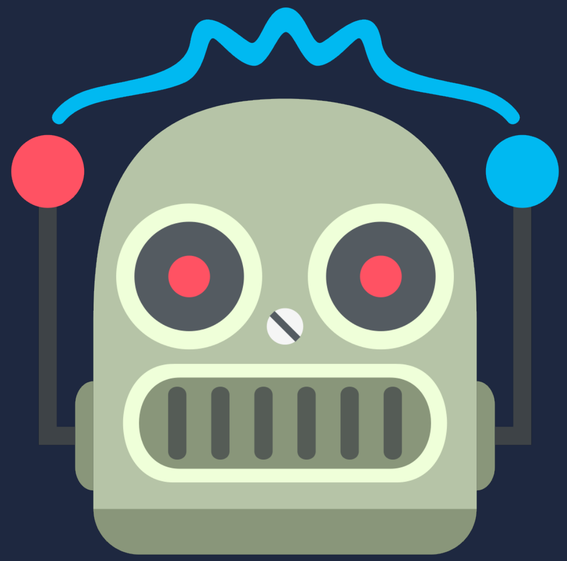


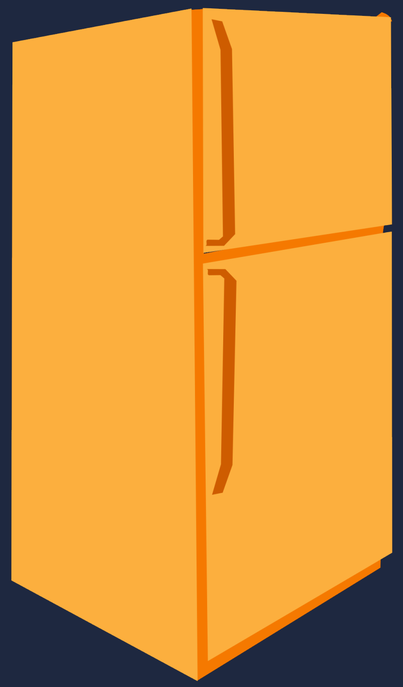
SimplyBuilt

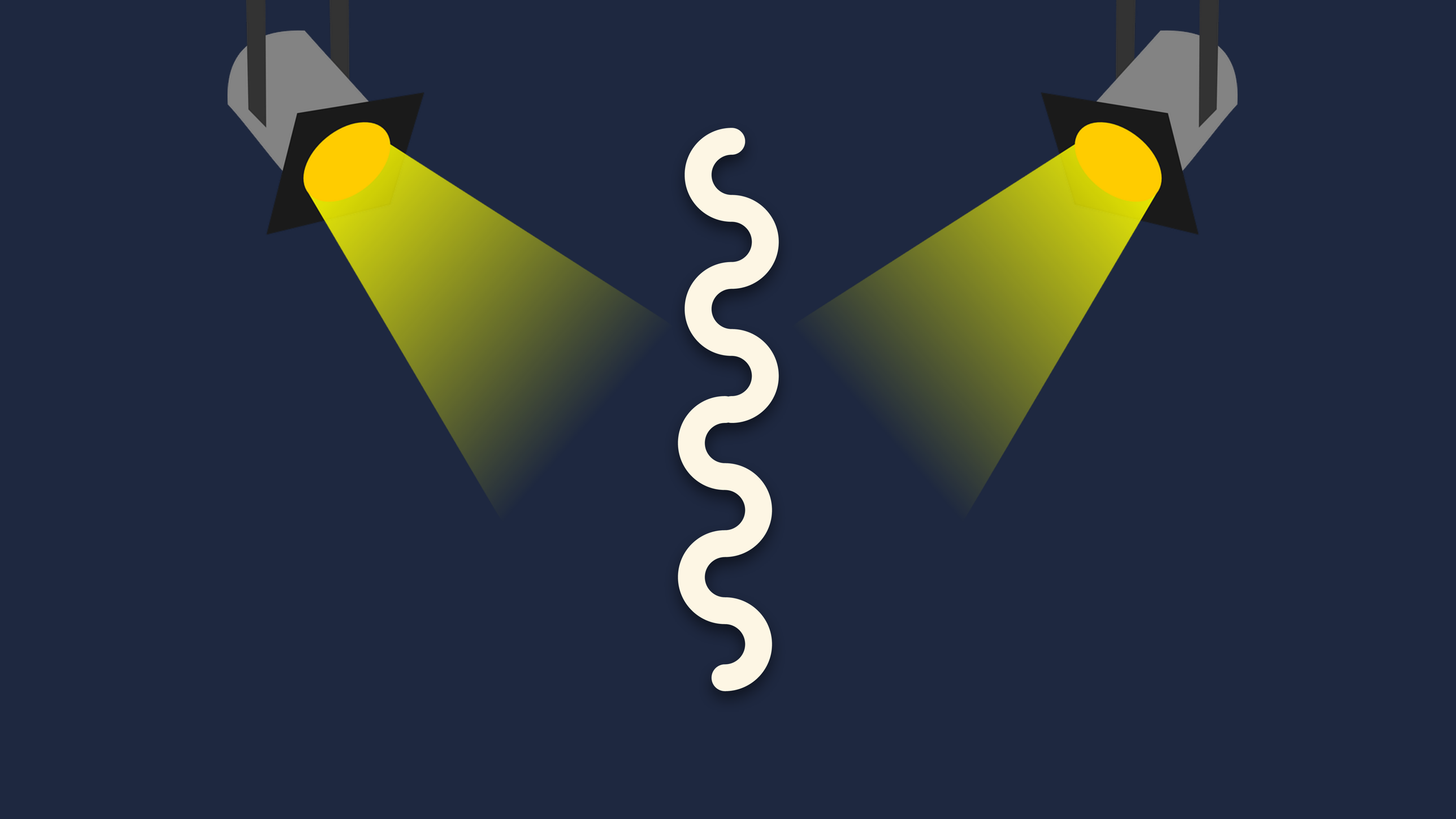
Your website, SimplyBuilt.

simplybuilt.com


JS







“Call me maybe?”

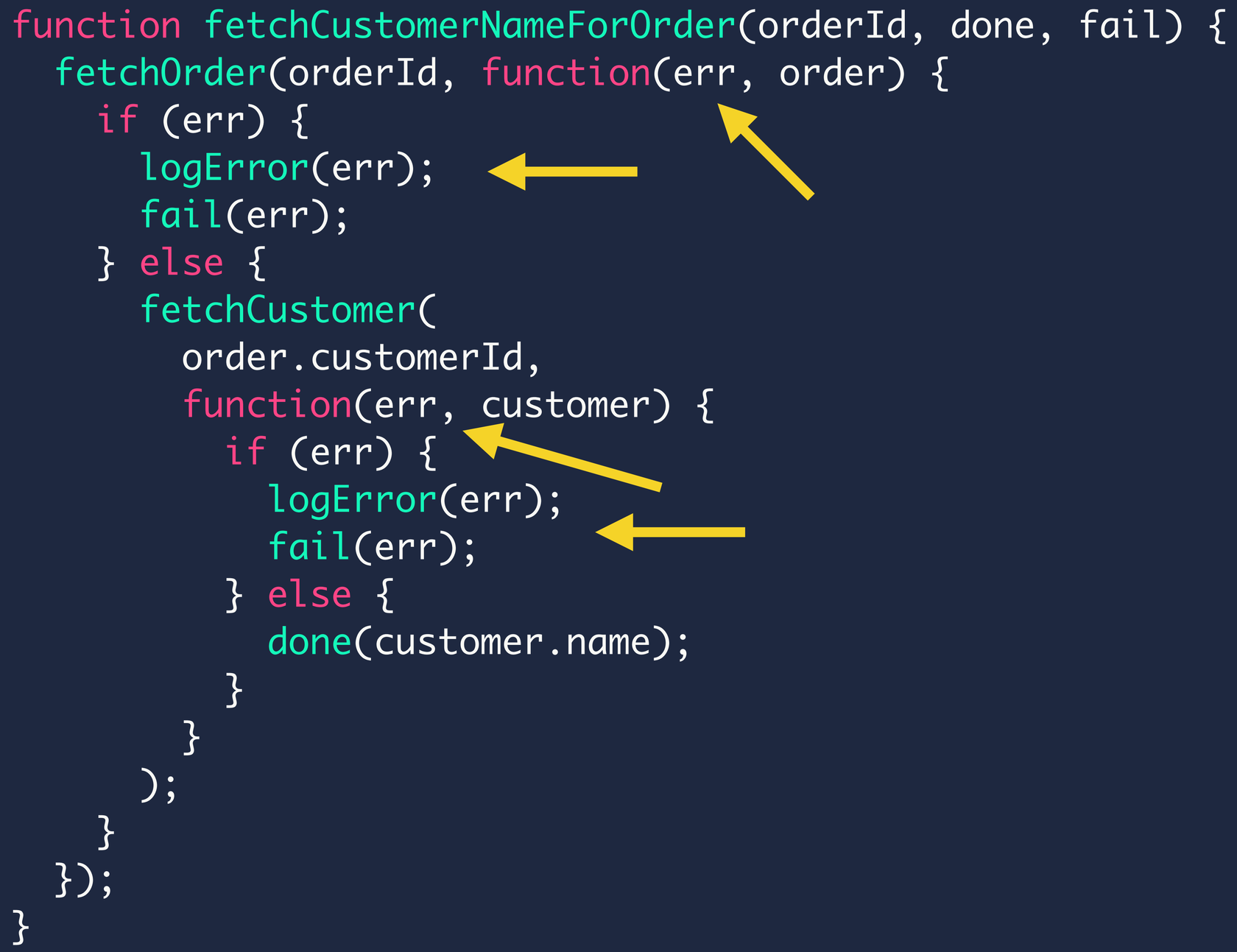
Async API


```
fetchUserById(1, function(err, user) {
  if (err) {
    console.error('Could not retrieve user');
  } else {
    console.log(user);
  }
});
```

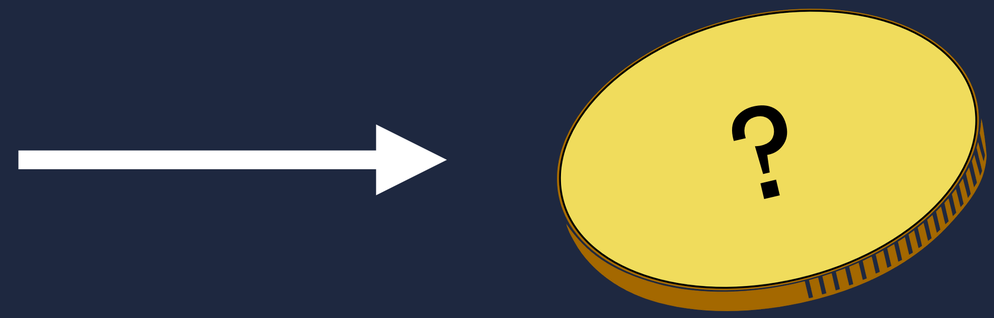

});
Callback
});
Callback
});
Callback
});
Callback
});
Callback
});
Callback

```
function fetchCustomerNameForOrder(orderId, done, fail) {
  fetchOrder(orderId, function(err, order) {
    if (err) {
      logError(err);
      fail(err);
    } else {
      fetchCustomer(
        order.customerId,
        function(err, customer) {
          if (err) {
            logError(err);
            fail(err);
          } else {
            done(customer.name);
          }
        }
      );
    }
  });
}
```

```
function fetchCustomerNameForOrder(orderId, done, fail) {
  fetchOrder(orderId, function(err, order) {
    if (err) {
      logError(err);
      fail(err);
    } else {
      fetchCustomer(
        order.customerId,
        function(err, customer) {
          if (err) {
            logError(err);
            fail(err);
          } else {
            done(customer.name);
          }
        }
      )
    }
  })
}
```


function fetchCustomerNameForOrder(orderId, done, fail) {
fetchOrder(orderId, function(err, order) {
if (err) {
logError(err);
fail(err);
} else {
fetchCustomer(
order.customerId,
function(err, customer) {
if (err) {
logError(err);
fail(err);
} else {
done(customer.name);
}
}
);
}
});
}

Async API

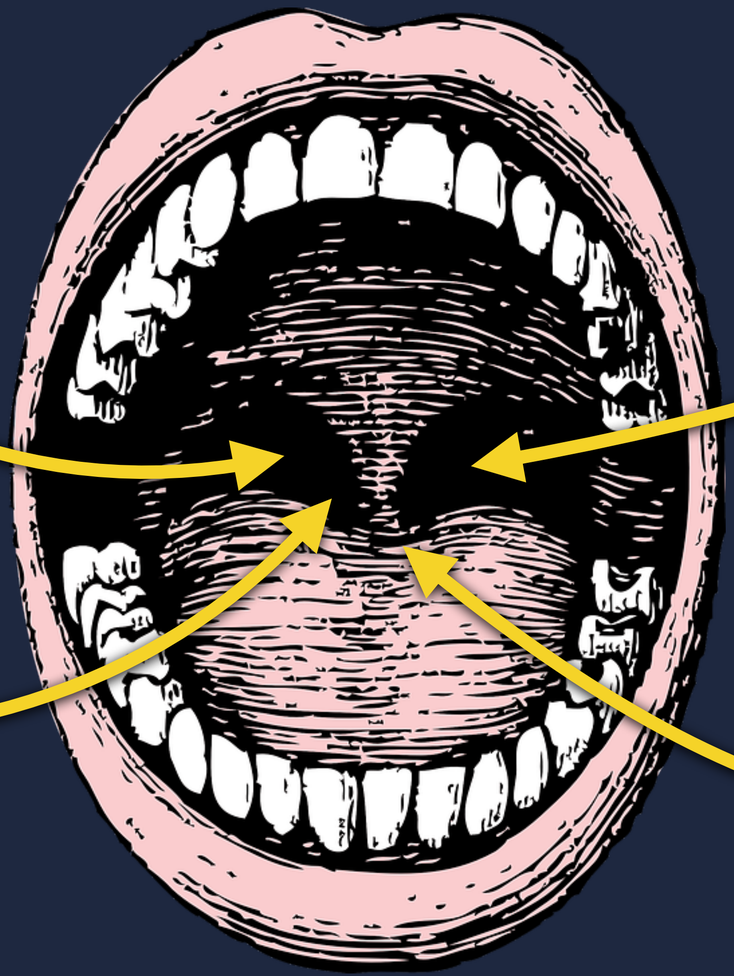


```
function fetchCustomerNameForOrder(orderId) {
  return fetchOrder(orderId)
    .then(order => fetchCustomer(order.customerId))
    .then(customer => customer.name);
}
```

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

Error

Error

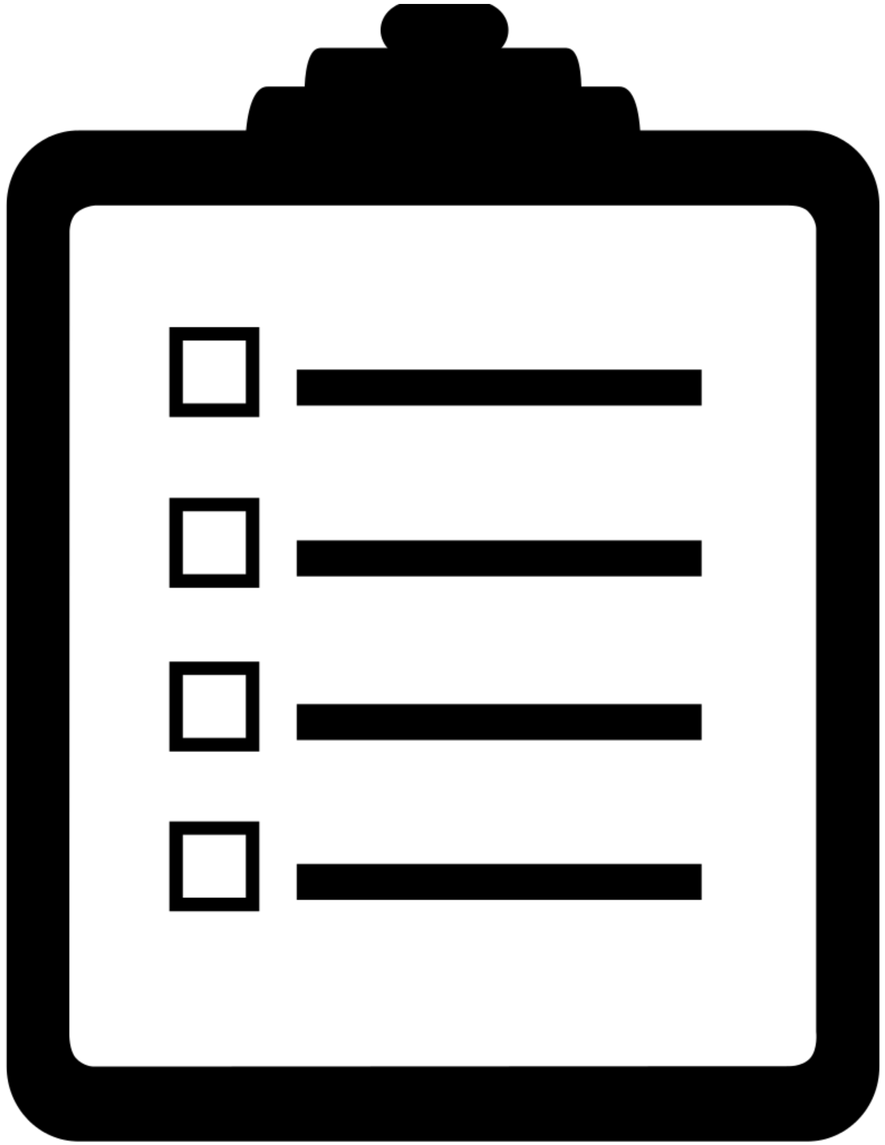


Error

Error

```
function fetchCustomerNameForOrder(orderId) {
  return fetchOrder(orderId)
    .then(order => fetchCustomer(order.customerId))
    .then(customer => customer.name)
    .catch(err => {
      logError(err);
      throw err;
    });
}
```

Getting there...

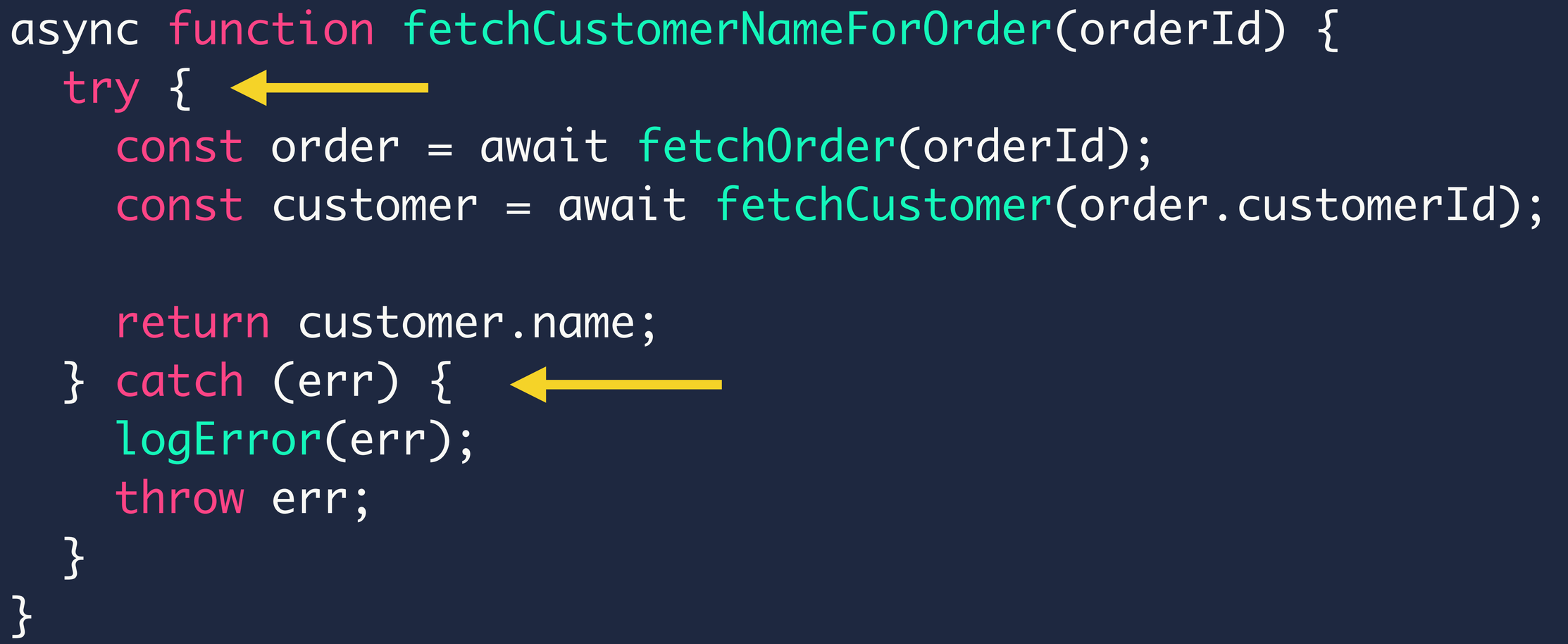


- Readable, synchronous program flow
- Nonblocking
- Use native flow control constructs

```
async function fetchCustomerNameForOrder(orderId) {
  try {
    const order = await fetchOrder(orderId);
    const customer = await fetchCustomer(order.customerId);

    return customer.name;
  } catch (err) {
    logError(err);
    throw err;
  }
}
```

```
async function fetchCustomerNameForOrder(orderId) {
  try {
    const order = await fetchOrder(orderId);
    const customer = await fetchCustomer(order.customerId);

    return customer.name;
  } catch (err) {
    logError(err);
    throw err;
  }
}
```

```
async function fetchCustomerNameForOrder(orderId) {
  try {
    const order = await fetchOrder(orderId);
    const customer = await fetchCustomer(order.customerId);

    return customer.name;
  } catch (err) {
    logError(err);
    throw err;
  }
}
```


async function fetchCustomerNameForOrder(orderId) {
try {
const order = await fetchOrder(orderId);
const customer = await fetchCustomer(order.customerId);
return customer.name;
} catch (err) {
logError(err);
throw err;
}
}

Await

```
function fetchOrder(orderId) {
  return fetch(`/orders/${orderId}`)
    .then(response => response.json());
}

function printOrder(orderId) {
  fetchOrder(orderId)
    .then(order => console.log(order));
}
```

```
function fetchOrder(orderId) {
  return fetch(`/orders/${orderId}`)
    .then(response => response.json());
}

async function printOrder(orderId) {
  const order = await fetchOrder(orderId);
  console.log(order);
}
```

fetchOrder(orderId)

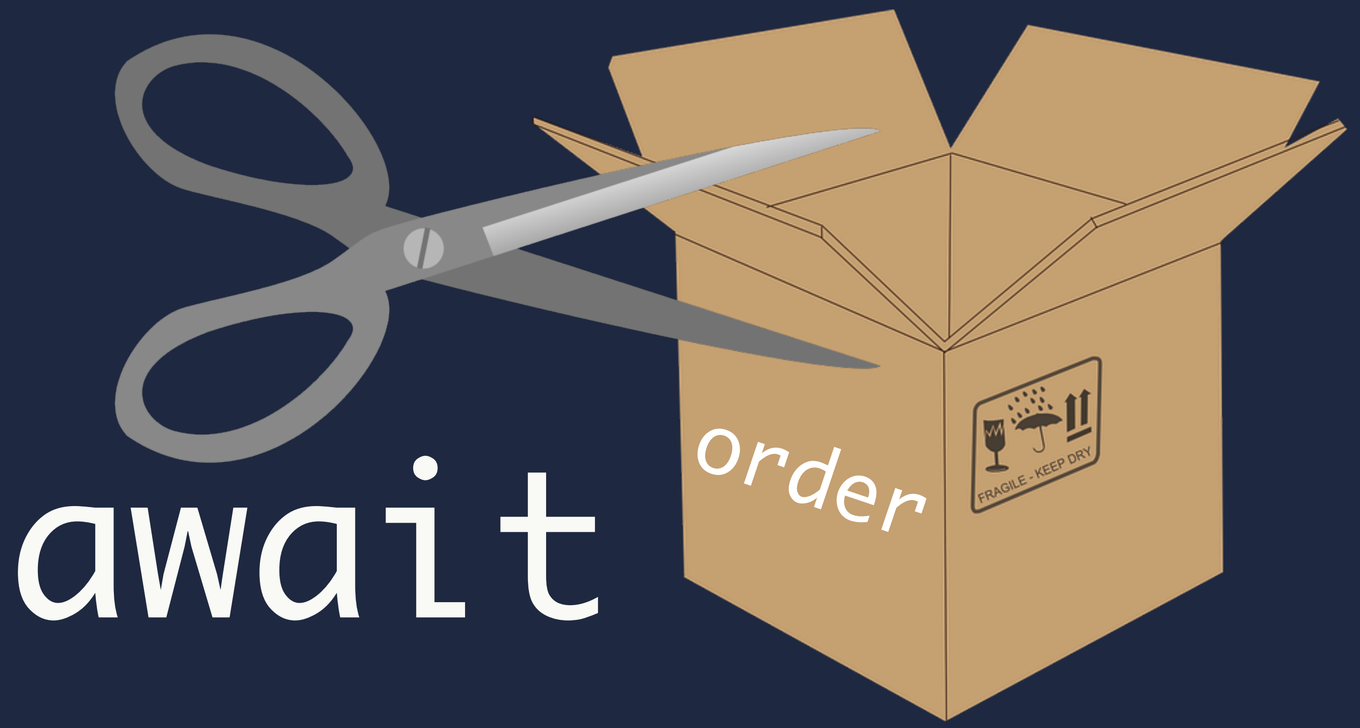

```
async printOrder(orderId) {
  const order = await fetchOrder(orderId);
  console.log(order);
}
```

await


```
async printOrder(orderId) {
  const order = await fetchOrder(orderId);
  console.log(order);
}
```

await


```
async printOrder(orderId) {
  const order = await fetchOrder(orderId);
  console.log(order);
}
```



await

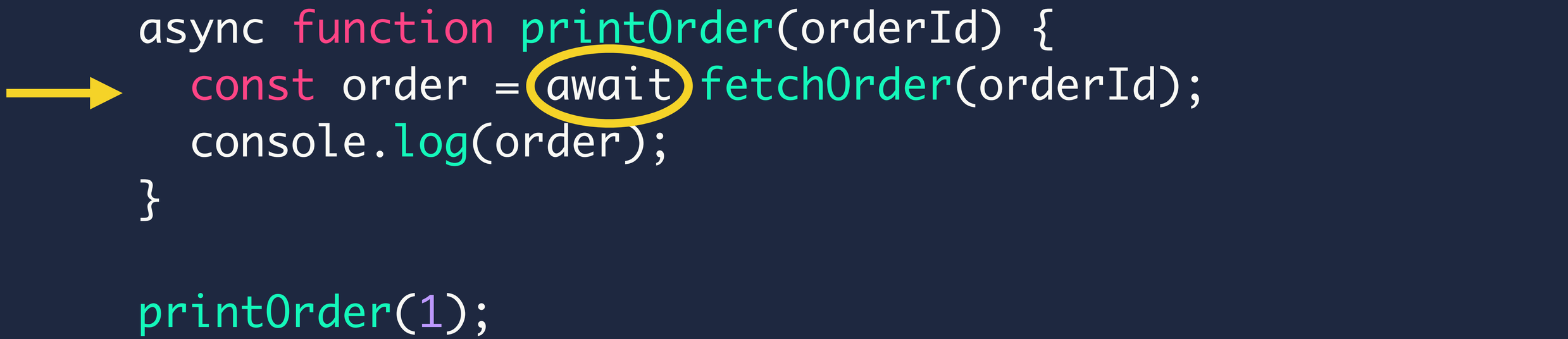
await

order

```
async printOrder(orderId) {
  const order = await fetchOrder(orderId);
  console.log(order);
}
```

```
async function printOrder(orderId) {
  const order = await fetchOrder(orderId);
  console.log(order);
}

printOrder(1);
```

Invoke



Encounter await

Wrap in `Promise.resolve`

```
async function printOrder(orderId) {
  const promise = Promise.resolve(fetchOrder(orderId));

  console.log(order);
}

printOrder(1);
```

Wrap in `Promise.resolve`

```
async function printOrder(orderId) {
  const promise = Promise.resolve(fetchOrder(orderId));

  console.log(order);
}

printOrder(1);
```

Wrap rest with then callback

```
async function printOrder(orderId) {
  const promise = Promise.resolve(fetchOrder(orderId));

  return promise.then(order => {
    console.log(order);
    return Promise.resolve(undefined);
  });
}

printOrder(1);
```

Wrap rest with then callback

```
async function printOrder(orderId) {
  const promise = Promise.resolve(fetchOrder(orderId));

  return promise.then(order => {
    console.log(order);
    return Promise.resolve(undefined);
  });
}

printOrder(1);
```

Wrap implicit return with Promise.resolve

```
function fetchCustomerNameForOrder(orderId) {
  return Promise.resolve(fetchOrder(orderId))
    .then(order => {
      return Promise.resolve(fetchCustomer(order.customerId))
        .then(customer => {
          return Promise.resolve(customer.name);
        });
    });
}
```

```
async function meaningOfLife() {
  return 42;
}

meaningOfLife() === 42;
```

???

```
async function meaningOfLife() {
  return 42;
}
```



```
function meaningOfLife() {
  return Promise.resolve(42);
}
```



```
meaningOfLife()
  .then(answer => answer === 42);
```

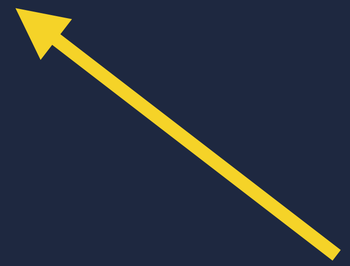
Exceptional Situations

```
async function printOrder(orderId) {
  try {
    const order = await fetchOrder(orderId);
    console.log(order);
  } catch (e) {
    console.log('Error retrieving order', e);
  }
}
```

```
async function printOrder(orderId) {
  try {
    const order = await fetchOrder(orderId);
    console.log(order);
  } catch (e) {
    console.log('Error retrieving order', e);
  }
}
```

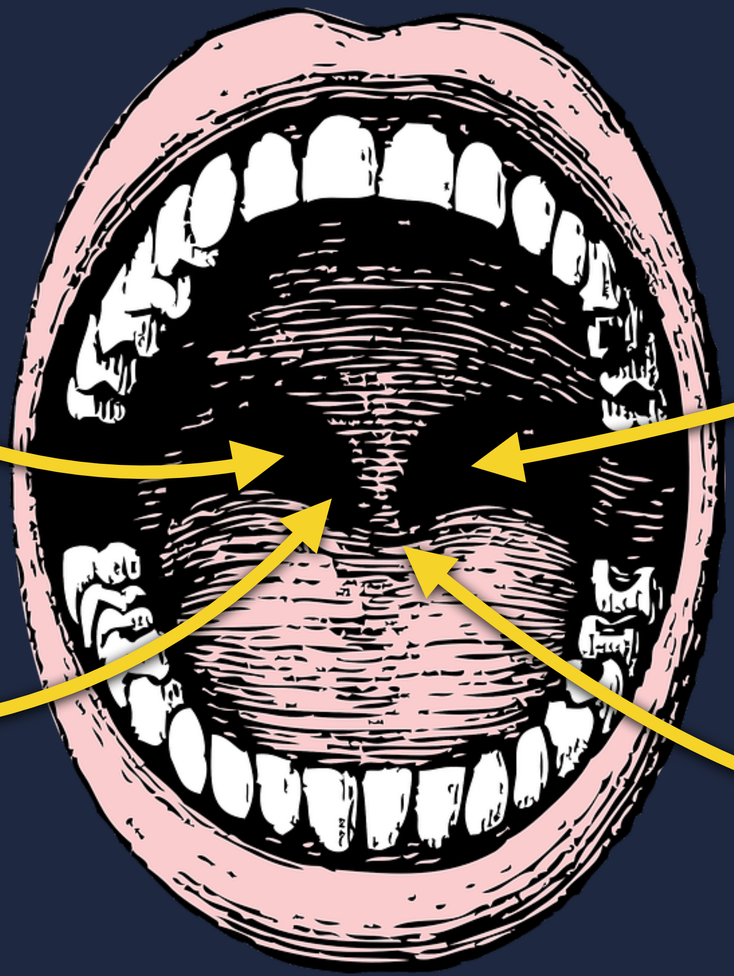
```
function fetchOrder(orderId) {
  return fetch(`/orders/${orderId}`)
    .then(resp => {
      if (resp.status === 404) {
        throw new Error('Order not found');
      }
      return resp.json();
    });
}
```

```
function fetchOrder(orderId) {
  return new Promise((resolve, reject) => {
    $.get(`/orders/${orderId}`)
      .done(order => resolve(order))
      .fail(resp => {
        if (resp.status === 404) {
          reject(new Error('Order not found'));
        }
      });
  });
}
```



Error

Error

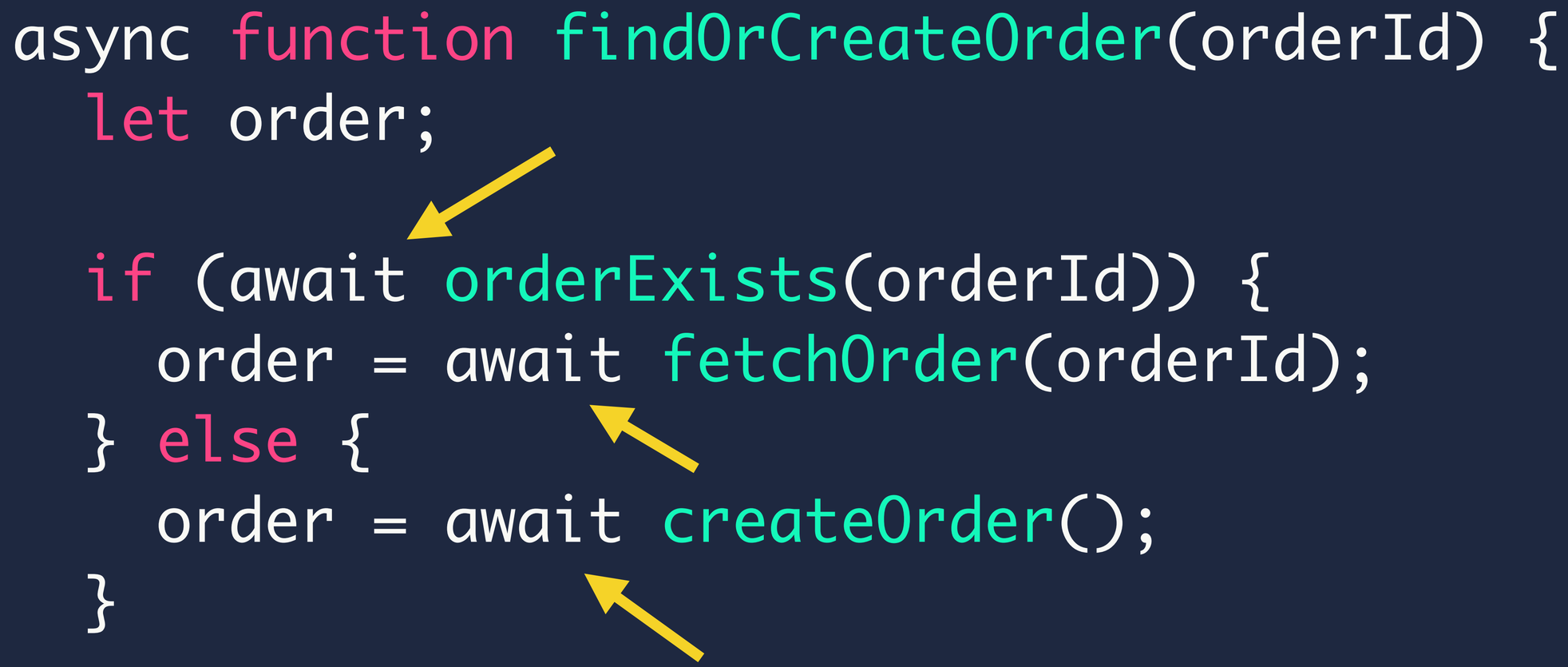


Error

Error

Regain Control

```
async function findOrCreateOrder(orderId) {
  let order;

  if (await orderExists(orderId)) {
    order = await fetchOrder(orderId);
  } else {
    order = await createOrder();
  }

  return order;
}
```


async function findOrCreateOrder(orderId) {
let order;
if (await orderExists(orderId)) {
order = await fetchOrder(orderId);
} else {
order = await createOrder();
}
return order;
}

```
async function printOrders(orderIds) {
  for (const id of orderIds) {
    const order = await fetchOrder(id);
    console.log(order);
  }
}
```

```
async function printOrders(orderIds) {
  for (const id of orderIds) {
    const order = await fetchOrder(id);
    console.log(order);
  }
}
```

```
async function printOrders(orderIds) {
  for (const id of orderIds) {
    const order = await fetchOrder(id);
    console.log(order);
  }
}
```

Problem?

```
async function printOrders(orderIds) {
  for (const id of orderIds) {
    const order = await fetchOrder(id);
    console.log(order);
  }
}

printOrders([1, 2, 3]);
```

```
printOrders([1, 2, 3]);
```

```
await fetchOrder(1);
await fetchOrder(2);
await fetchOrder(3);
```

```
printOrders([1, 2, 3]);
```

```
await fetchOrder(1);
await fetchOrder(2);
await fetchOrder(3);
```

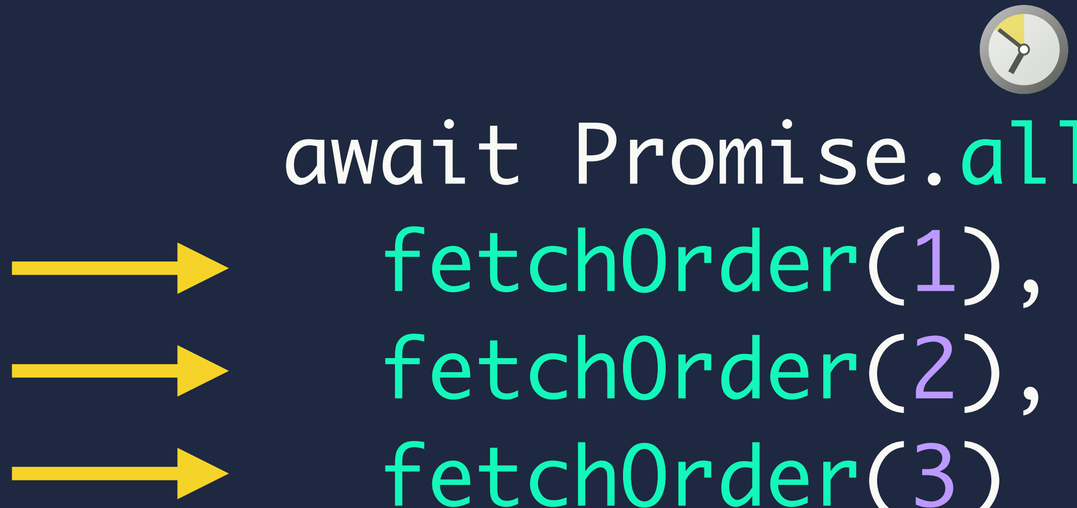
```
printOrders([1, 2, 3]);
```

```
await fetchOrder(1);
await fetchOrder(2);
await fetchOrder(3);
```

```
async function printOrders(orderIds) {
  const orders = await Promise.all(
    orderIds.map(fetchOrder)
  );

  orders.forEach(order => console.log(order));
}
```

```
async function printOrders(orderIds) {
  const orders = await Promise.all(
    orderIds.map(fetchOrder)
  );

  orders.forEach(order => console.log(order));
}
```

```
printOrders([1, 2, 3]);

await Promise.all([
  fetchOrder(1),
  fetchOrder(2),
  fetchOrder(3)
]);
```



```
printOrders([1, 2, 3]);
```

Demo

rise.of.async.jeremyfairbank.com

BABEL

```
$ npm install --save-dev \
    babel-core \
    babel-cli \
    babel-preset-es2015 \
    babel-plugin-syntax-async-functions \
    babel-plugin-transform-regenerator \
    babel-plugin-transform-runtime
```

.babelrc

```
{
  "presets": ["es2015"],
  "plugins": [
    "syntax-async-functions",
    "transform-regenerator",
    "transform-runtime"
  ]
}
```

```
$ node_modules/.bin/babel-node myAsyncFile.js
```

Resources

- tc39.github.io/ecmascript-asyncawait/
- github.com/tc39/ecmascript-asyncawait
- github.com/tc39/ecma262

THANKS!

CODE:

github.com/jfairbank/rise-of-async-js-talk

SLIDES:

speakerdeck.com/jfairbank/fluent-conf-rise-of-async-javascript

Jeremy Fairbank

blog.jeremyfairbank.com

@elpapapollo | gh/jfairbank