Loading Visual

When you try to sign in, create a user, or reset your password, a request is sent to Firebase. Usually, that request is pretty fast, but no matter the speed, let's give our users some feedback by showing a loading visual to them which also enhances the UI/UX.

We'll cover the following

Loading Visual HTML

```html
<!-- Loading visual cue -->
<div id="loading-outer-container">
  <div id="loading-inner-container">
    <div id="loadspin">
      <div id="circle1"></div>
      <div id="circle2"></div>
    </div>
  </div>
</div>
```

HTML

Create the `loading` Function

```javascript
// Function to hide and show the loading visual cue
loading = (action) => {
  if (action === 'show'){
      document.getElementById('loading-outer-container').style.display = 'block'
  } else if (action === 'hide' ){
      document.getElementById('loading-outer-container').style.display = 'none'
  } else {

      console.log('loading error')
  }
}
```

JavaScript

Modify Each Form Submit Event Listener

Invoke `loading('show')`

We invoke the `loading('show')` function as the submit event happens.

Invoke `loading('hide')`

We invoke the `loading('hide')` function in the `.catch()` of Firebase auth methods inside each submit event.

```
// Create user form submit event
createUserForm.addEventListener('submit', event => {
  event.preventDefault()
  loading('show')
  // Grab values from form
  const displayName = document.getElementById('create-user-display-name').value
  const email = document.getElementById('create-user-email').value
  const password = document.getElementById('create-user-password').value
  // Send values to Firebase
  auth.createUserWithEmailAndPassword(email, password)
    .then(() => {
      auth.currentUser.updateProfile({
        displayName: displayName
      })
      createUserForm.reset()
    })
    .catch(error => {
      displayMessage('error', error.message)
      loading('hide')
    })
})

// Sign in form submit event
signInForm.addEventListener('submit', event => {
  event.preventDefault()
  loading('show')
        // Grab values from form
  const email = document.getElementById('sign-in-email').value
        const password = document.getElementById('sign-in-password').value
        // Send values to Firebase
        auth.signInWithEmailAndPassword(email, password)
    .then(() => {
     signInForm.reset()
     hideAuthElements()
    })
    .catch(error => {
      displayMessage('error', error.message)
      loading('hide')
    })

})

// Forgot password form submit event
forgotPasswordForm.addEventListener('submit', event => {
  event.preventDefault()
  loading('show')
        // Grab value from form
        var emailAddress = document.getElementById('forgot-password-email').value
        // Send value to Firebase
    firebase.auth().sendPasswordResetEmail(emailAddress)
    .then(() => {
      forgotPasswordForm.reset()
      displayMessage('success', 'Message sent. Please check your email')
     })
    .catch(error => {
      displayMessage('error', error.message)
      loading('hide')
    })
})
```

JavaScript

Modify the `hideAuthElements` function

Invoke `loading('hide')`

We invoke the `loading('hide')` function in the `hideAuthElements()` function.

```
hideAuthElements = () => {
  clearMessage()
  loading('hide')
  createUserForm.classList.add('hide')
  signInForm.classList.add('hide')
  forgotPasswordForm.classList.add('hide')
  createUserDialog.classList.add('hide')
  signInDialog.classList.add('hide')
  haveOrNeedAccountDialog.classList.add('hide')
}
```

JavaScript

Loading CSS

In this course, including CSS has usually been optional. However, in this lesson, it is mandatory. Without the CSS code, you will not see a loading animation.

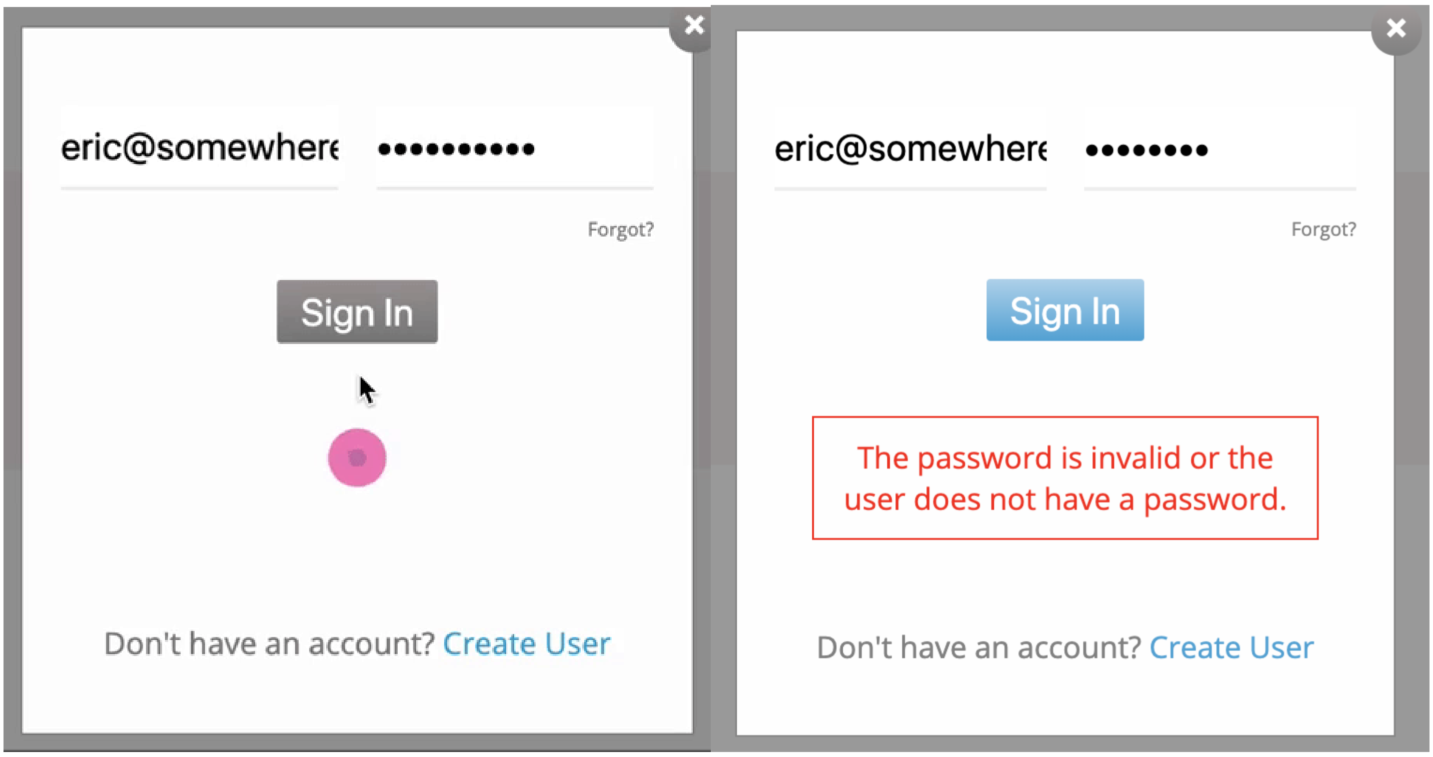
```
/* Loading visual Lesson */
#loading-outer-container{
    margin: 40px auto;
}
```

```
#loading-inner-container{
    position:relative;
    width:40px;

    height:40px;
    left:0;
    right:0;
    margin:auto;
    top:20%;
}

#loadspin{
    width:40px;
    height:40px;
    position:absolute;
    margin:0 auto;
}

#circle2{
    background-color:#39B1C6
}

#circle1{
    background-color:#EC3F8C
}

#circle1, #circle2{
    width:100%;
    height:100%;
    border-radius:50%;
    opacity:.6;
    position:absolute;
    top:0;
    left:0;
    -webkit-animation:bounce 2s infinite ease-in-out;
    animation:bounce 2s infinite ease-in-out
}

#circle2{
    -webkit-animation-delay:-1s;
    animation-delay:-1s
}

@-webkit-keyframes bounce {
    0%,100%{
        -webkit-transform:scale(0)
    }
    50%{
        -webkit-transform:scale(1)
    }
}

@keyframes bounce{
    0%,100% {
        transform:scale(0);
        -webkit-transform:scale(0)
    }

    50% {
        transform:scale(1);
        -webkit-transform:scale(1)
    }
}
```

CSS

Expected Output

Below on the left, you can see what the loading visual will look like. On the right is the message from the server. Once the message is present, the loading visual is hidden.

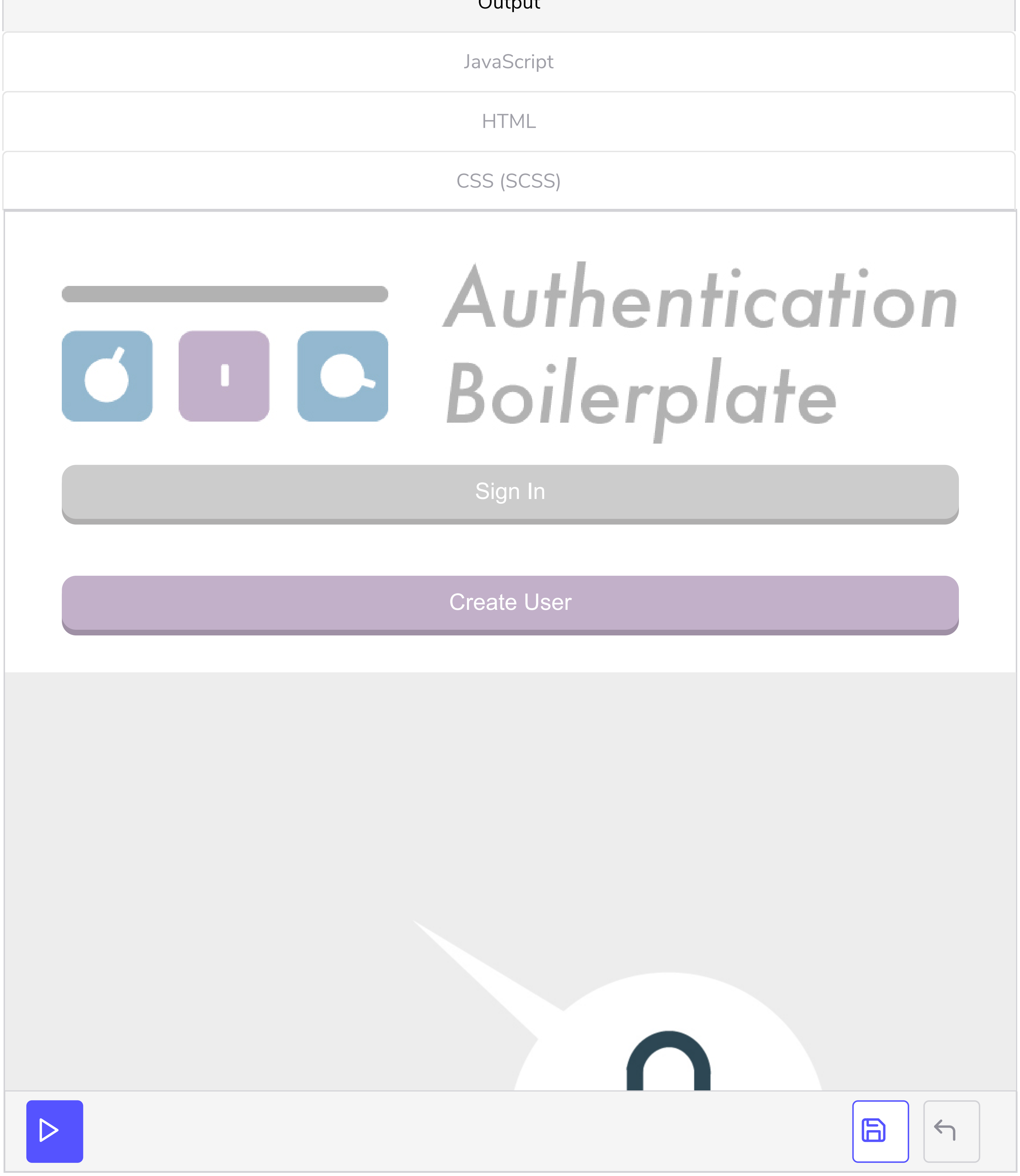

eric@somewhere
Forgot?
Sign In
Don't have an account? Create User
eric@somewhere
Forgot?
Sign In
The password is invalid or the user does not have a password.
Don't have an account? Create User

The Authentication Boilerplate Application

Firebase Authentication goes so fast that trying to catch a glimpse of the loading visual might be hard, especially if you have a decent internet connection. Watch closely though, as you might see it for a split second before firebase sends a response and it gets hidden again.

This code requires the following API keys to execute:

apiKey	Not Specified...
authDomain	Not Specified...
databaseURL	Not Specified...
projectId	Not Specified...
storageBucket	Not Specified...
messagingSenderId	Not Specified...
appId	Not Specified...



In the next lesson, you will be quizzed on what you have learned about Firebase authentication.