codecademy

Session Authentication in Express.js

Session

A **session** is a storage strategy that consists of information server-side.
A **session id**, as well as other session variables, are stored client-side in cookies or `localStorage` and allow the browser to make an HTTP request to get the persistent session information from the server.
Sessions are terminated when a user exits the browser or after client storage is cleared.

Sessions: Cookies

A **cookie** is a text file that stores stateful client data in a key-pair format.
It is stored by the web browser, aka client-side. Cookies are first defined by the server, then stored by the browser. The cookie may be sent in requests to the server.
Session cookies keep track of the session ID and expire automatically if there's a time provided or when the browser closes. One cookie for a domain can store 4kb.

```
Set-Cookie: Key=Value
```

Sessons: localStorage and sessionStorage

`localStorage` and `sessionStorage` are client-side browser storage introduced with HTML5 that stores data in a key-pair format. It *does not* interact with the server and is only changeable through JavaScript, with simple syntax.
`localStorage` persists after a user exits the browser, while sessionStorage clears when the browser is exited.
Per domain, `localStorage` can store 10mb while `sessionStorage` can store 5mb.

HTTP Security Headers Definition

HTTP Security Headers are HTTP Response Headers that change how a client's browser will behave when handling your web content. They are name-value pairs in added

server-side and provide additional security for a web application and a user.
Here's an example:

```
Strict-Transport-Security: max-age=31536000; i
```

HTTP Security Headers: Strict-Transportation-Security

The Strict-Transportation-Security HTTP security header enforces the use of HTTPS when a browser accesses a website.
Here's an example of the Strict-Transport-Security header:

```
Strict-Transport-Security: max-age=31536000; i
```

The `includeSubDomains` value tells the browser that the current site, including all of its sub-domains, is HTTPS-only.
The `max-age` field tells the browser to remember this for the next year (31536000 seconds = 1 year), reducing redirect responses to the HTTPS version of the site in the future.

HTTP Security Headers: Content-Security-Policy

The `Content-Security-Policy` HTTP security header restricts where content-like scripts can load from, which helps prevent Cross-Site-Scripting (XSS) attacks. Options for the header include: only loading from the same domain (`self`), specifying for a type of content, and more.
Here's an example of the `Content-Security-Policy` header:

```
Content-Security-Policy: script-src 'self';
```

The `script-src` option restricts which resources JavaScript can be loaded from. The `self` value indicates that the browser should only run scripts from the current domain.

HTTP Security Headers: X-Frame-Options

Here are some examples of the `X-Frame-Options` header that help prevent clickjacking (sneaky iframes) attacks.

```
X-Frame-Options: DENY
```

means your page can't be hidden in an iframe anywhere,

whereas
`X-frame-Option: SAMEORIGIN`

only allows this page to be put into an iframe within your own domain.
`X-Frame-Options: ALLOW-FROM https://example.co`

lets you list sites that are allowed to put the current content in an iframe.

Sessions in Express.js

The express-session module can be used as a middleware to implement sessions in Express.js. A full configuration looks something like this:

```
app.use(
 session({
   secret: "qEas5ns3gxl41G",
   cookie: { maxAge: 86400000, secu
   resave: false,
   saveUninitialized: false,
   store
 })
);
```

where the session secret is used to encrypt the session data stored in the browser, the cookie expiration and options are set, and the `resave`, `saveUninitialized` booleans are set.

express-session-data

In express-session, `req.session` is used to access the session associated with a request. Session data is serialized as JSON and can be stored in memory, in database, or in a memory cache.
The syntax for finding something in a session object looks something like this: `req.session.user.username`.

JSON Web Token (JWT)

A JSON Web Token (JWT) is a self-contained JSON object that compactly and securely transmits information

`Authorization: Bearer`

between two parties. It is digitally signed using a secret or a public/private key pair.
It is often sent in the Authorization header using the Bearer schema, as shown in the code example.

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzd
WIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkhhcmluZSB
Db29wZXIiLCJhZG1pbiI6ZmFsc2UsImlhdCI6MTYyM
DkyNDQ3OCwiZXhwIjoxNjIwOTM5MTg3fQ.3B-
FLgPETrExxlDKW30AoU7KGE6xuZodw79TQR8_mwM
```

JWT Composition

LWBAT explain that a JWT consists of:

- A Header
- A Payload
- A Signature.

The header and payload will be Base64UrL encoded, and then each part is separated by a `.`, which looks like: `encodedHeader.encodedPayload.signature`.
A fully created JWT is shown in the code example.

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzd
WIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkhhcmluZSB
Db29wZXIiLCJhZG1pbiI6ZmFsc2UsImlhdCI6MTYyM
DkyNDQ3OCwiZXhwIjoxNjIwOTM5MTg3fQ.3B-
FLgPETrExxlDKW30AoU7KGE6xuZodw79TQR8_mwM
```

JWT Header

The JWT header contains:

- The Type "JWT"
- The Hashing Algorithm Used

Hashing algorithms might be something like HMAC SHA256 or RSA.

```
{
    "alg" : "HS256"
    "typ" : "JWT"
}
```

JWT Payload

A JWT Payload contains reserved, public, or private statements about the entity (aka "claims").

- *Reserved Claims* are predefined claims.
- *Public Claims* are defined by a developer and should be defined in the IANA JSON WEB Token Registry to avoid collisions.
- *Private Claims* are custom claims created to share information between two consenting parties.

```
{
  "sub": "555123456",
  "name": "Jane Doe",
  "admin": false
}
```

JWT Signature

A JWT Signature is created from the encoded header, encoded payload, a secret, and the algorithm specified.

```
HMACSHA256(
  base64UrlEncode(header) + "." +
  base64UrlEncode(payload),
  secret)
```

Save Print Share

codecademy