



I Don't Care About Security

And Neither Should You

About Me



@joel__lord



joellord



99¢ FRESH PIZZA

2 SLICE & 1 CAN OR WATER \$ 2.75

PEPPERONI • MUSHROOM
SAUSAGE • EXTRA CHEESE
BLACK OLIVE • PINEAPPLE

SPECIALS!

GET 2 CHEESE SLICES
& CAN SODA
OR BOTTLE WATER

18" PIE

\$2.75

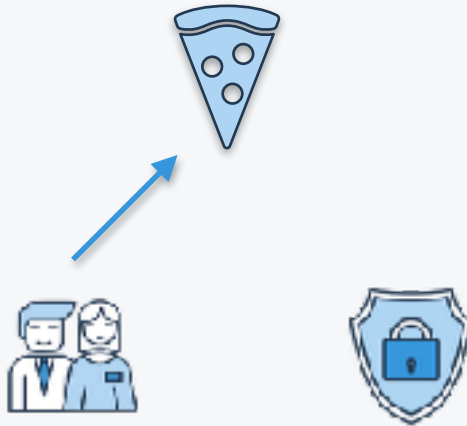
\$2.99

**FOR
RENT**



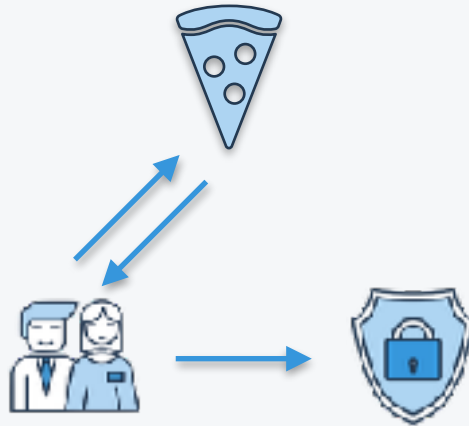
OAuth - Flows

Authorization Code



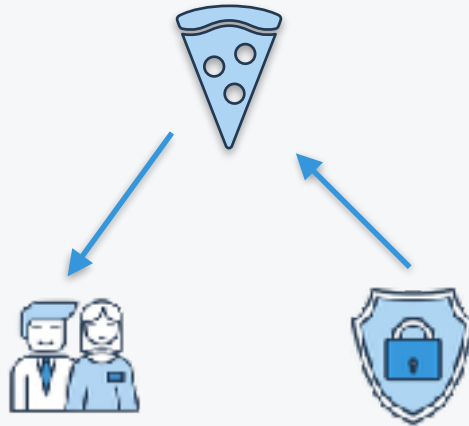
OAuth - Flows

Authorization Code



OAuth - Flows

Authorization Code



That reminds me of OAuth!

OAuth - Flows

Authorization Code



OAuth - Flows

Authorization Code



@joel__lord
#iJS18

OAuth - Flows

Authorization Code



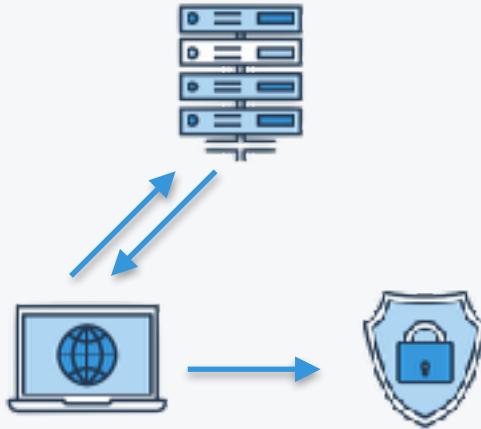
OAuth - Flows

Authorization Code



OAuth - Flows

Authorization Code



OAuth - Flows

Authorization Code



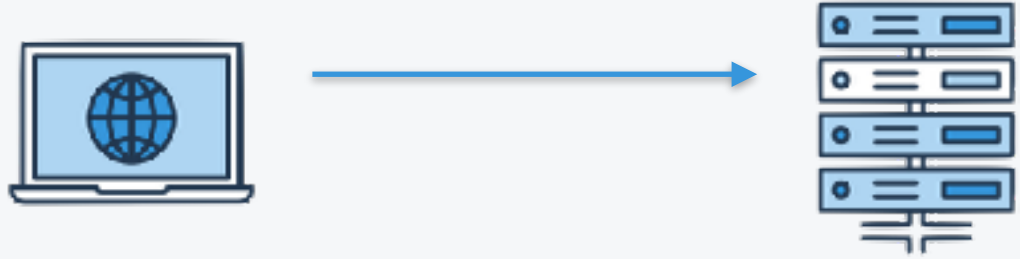
But Why?

A close-up, high-contrast photograph of a hockey stick and puck on an ice rink. The stick is positioned diagonally across the upper right portion of the frame, with its blade resting on the ice. A black puck is placed on the ice just in front of the stick's blade. The ice surface is covered in numerous scratches and scuffs, indicating frequent use. The lighting is dramatic, casting soft shadows and highlighting the textures of the ice and the stick.

Delegation!

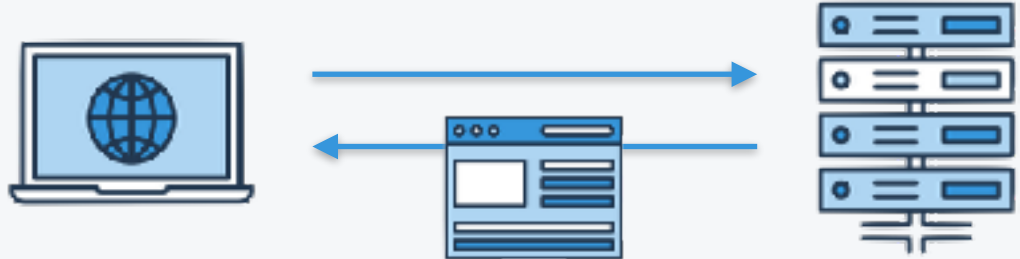
Traditional Applications

- Browser requests a login page



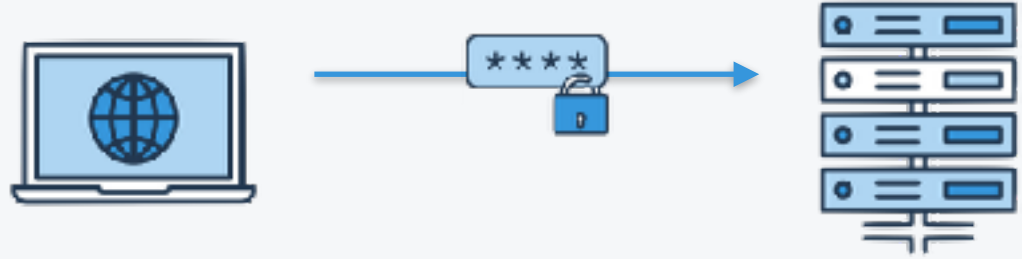
Traditional Applications

- Browser requests a login page



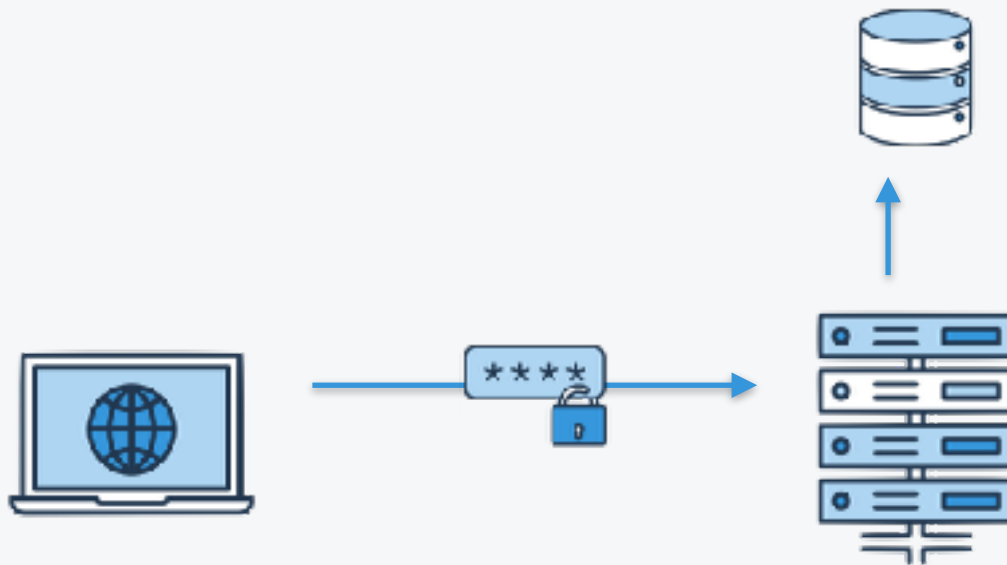
Traditional Applications

- Browser requests a login page



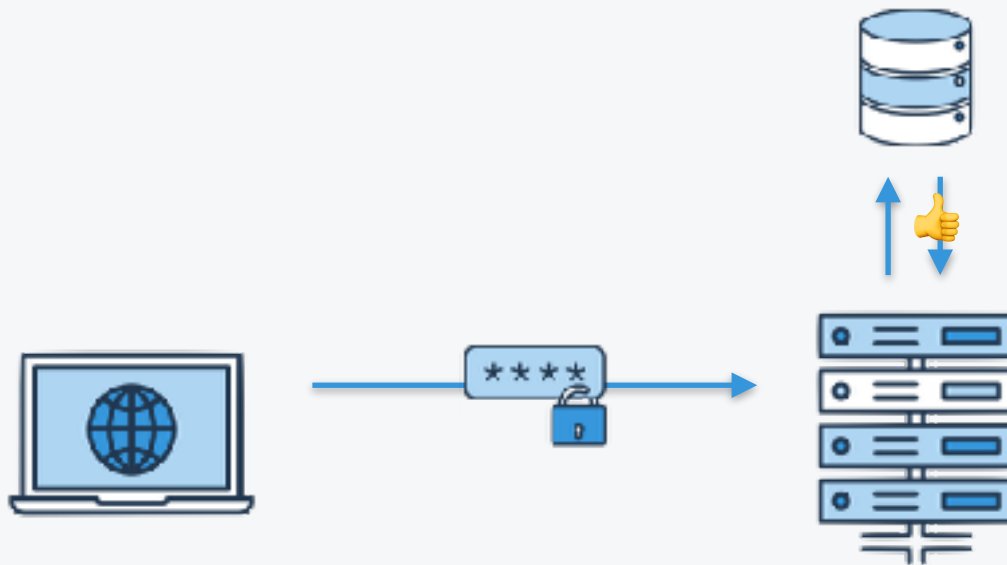
Traditional Applications

- Browser requests a login page
- The server validates on its database



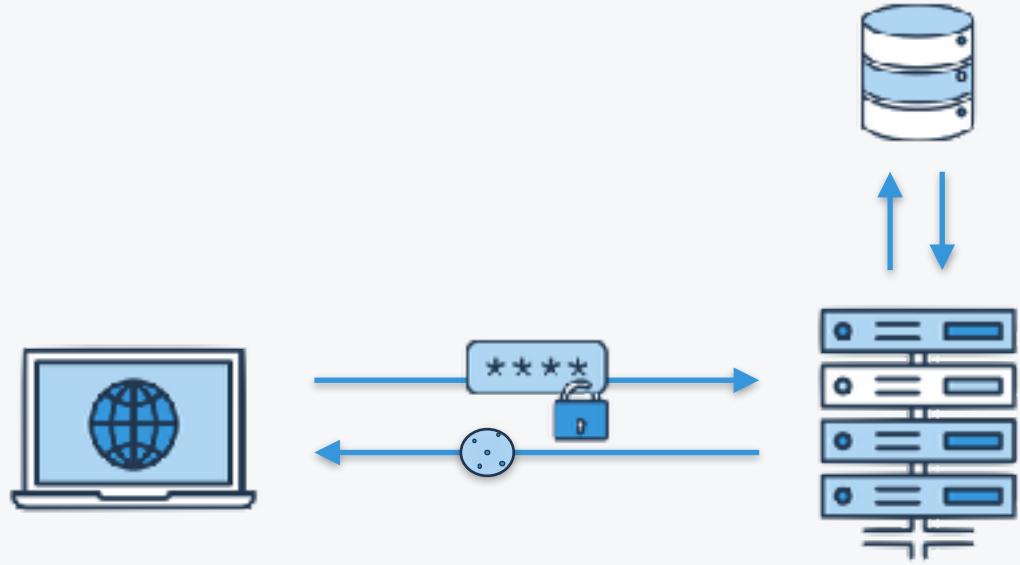
Traditional Applications

- Browser requests a login page
- The server validates on its database



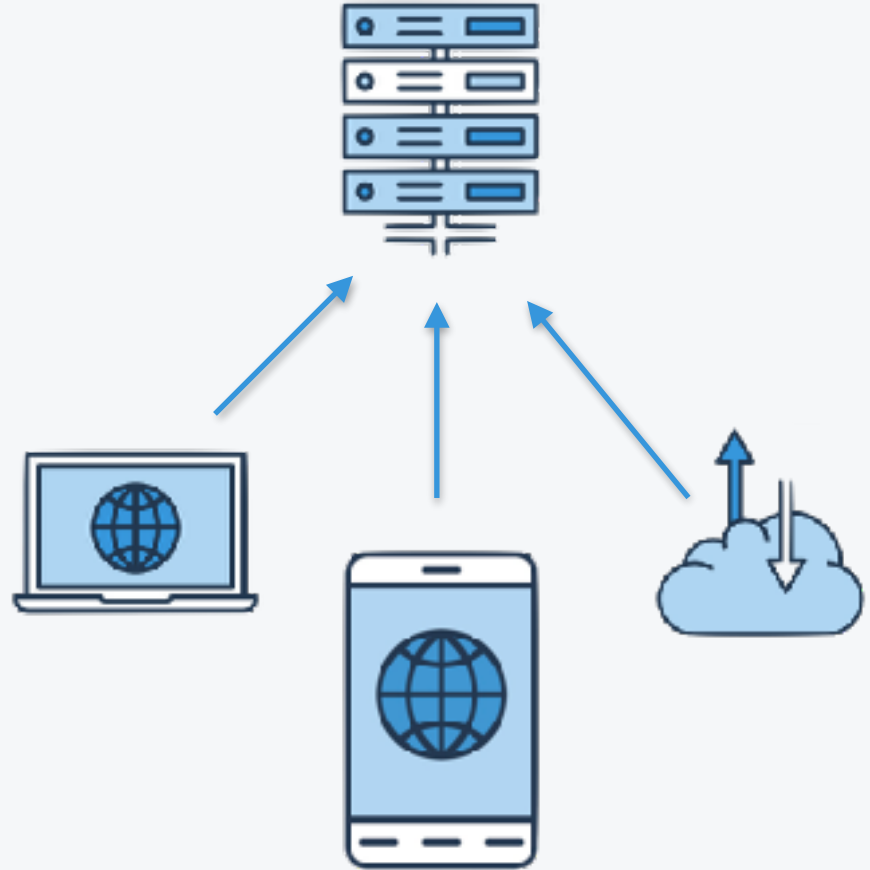
Traditional Applications

- Browser requests a login page
- The server validates on its database
- It creates a session and provides a cookie identifier



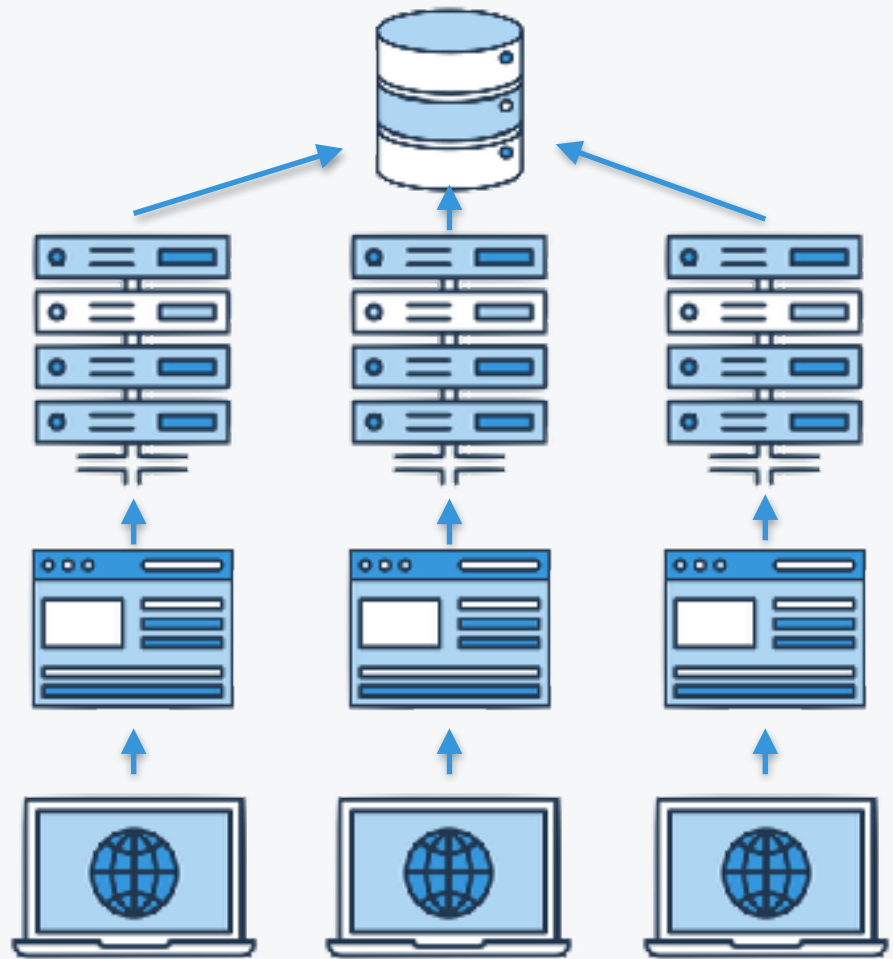
What's wrong with traditional auth?

- Multiple platforms connecting to your application



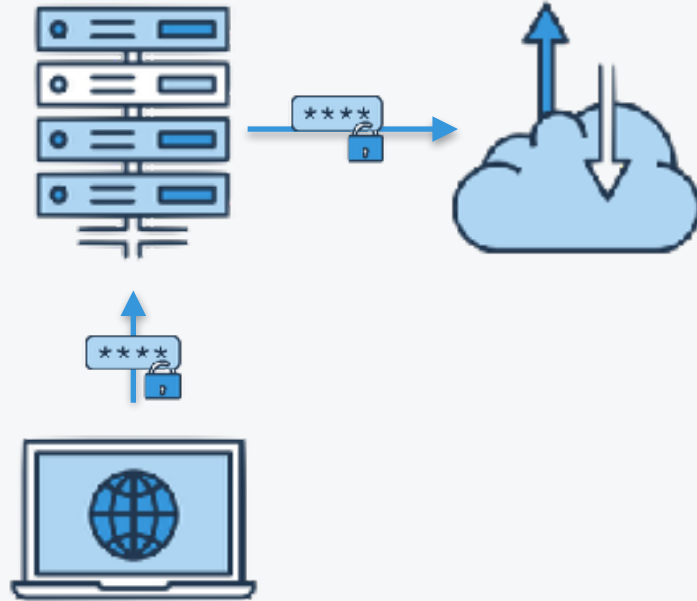
What's wrong with traditional auth?

- Multiple platforms connecting to your application
- Tightly coupled



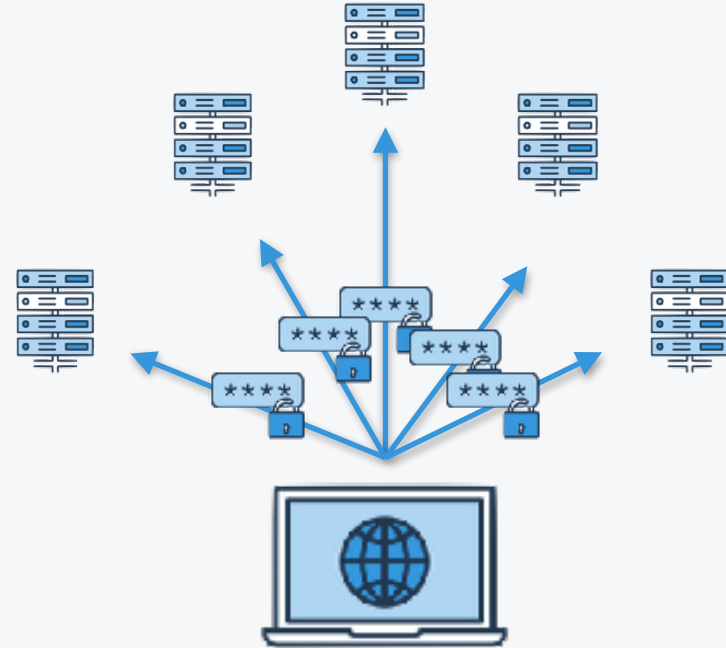
What's wrong with traditional auth?

- Multiple platforms connecting to your application
- Tightly coupled
- Sharing credentials to connect to another API



What's wrong with traditional auth?

- Multiple platforms connecting to your application
- Tightly coupled
- Sharing credentials to connect to another API
- Users have a gazillion passwords to remember, which increases security risks



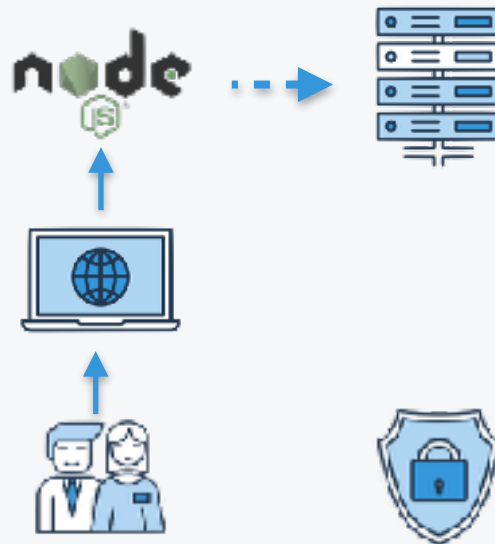


OAuth - The Flows

Authorization Code

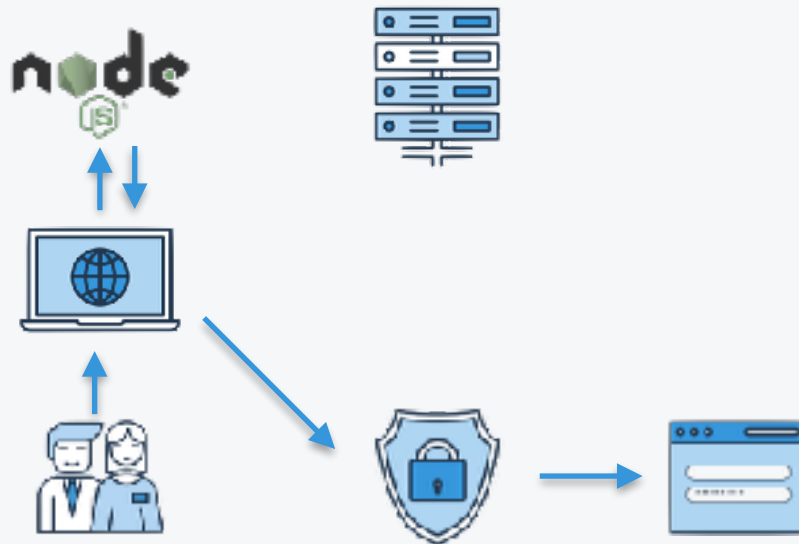
Authentication Flows

Authorization Code



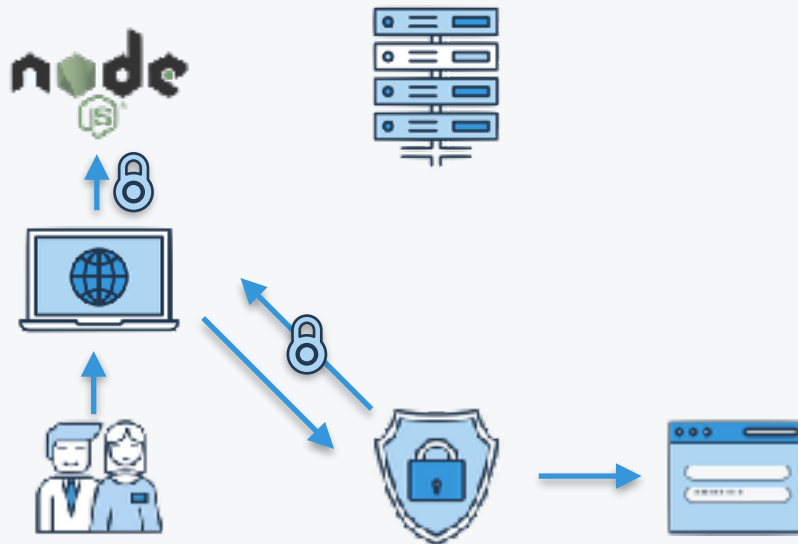
Authentication Flows

Authorization Code



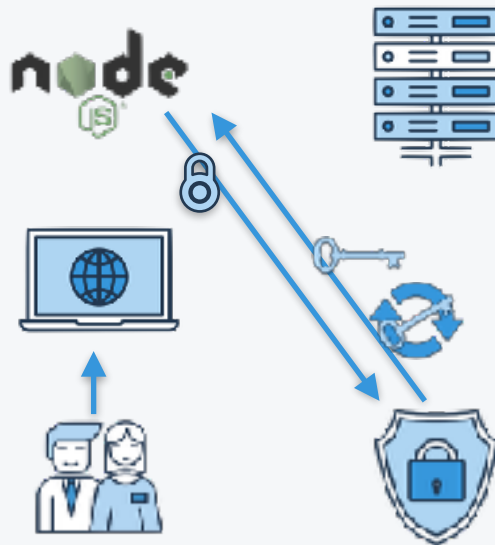
Authentication Flows

Authorization Code



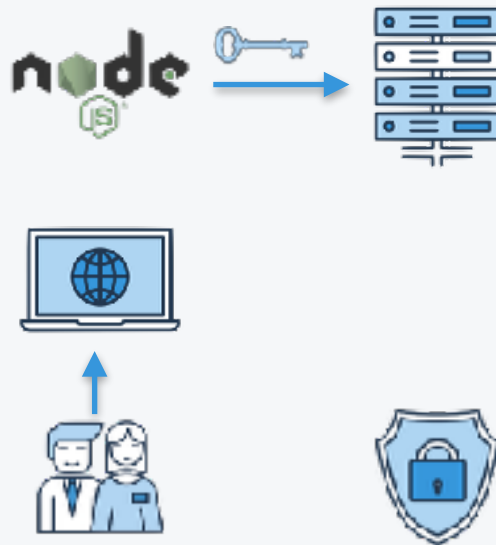
Authentication Flows

Authorization Code



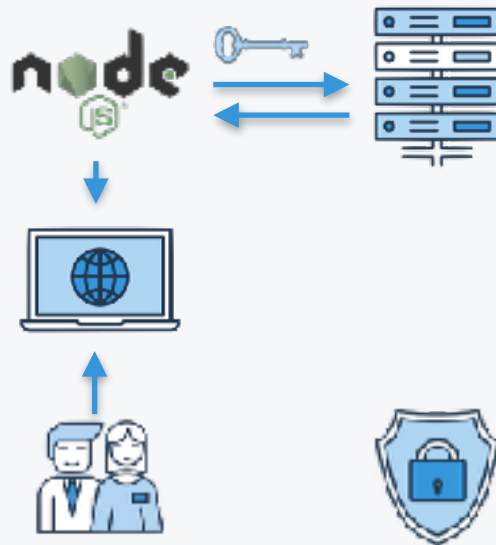
Authentication Flows

Authorization Code



Authentication Flows

Authorization Code



OAuth - The Flows

Implicit Flow

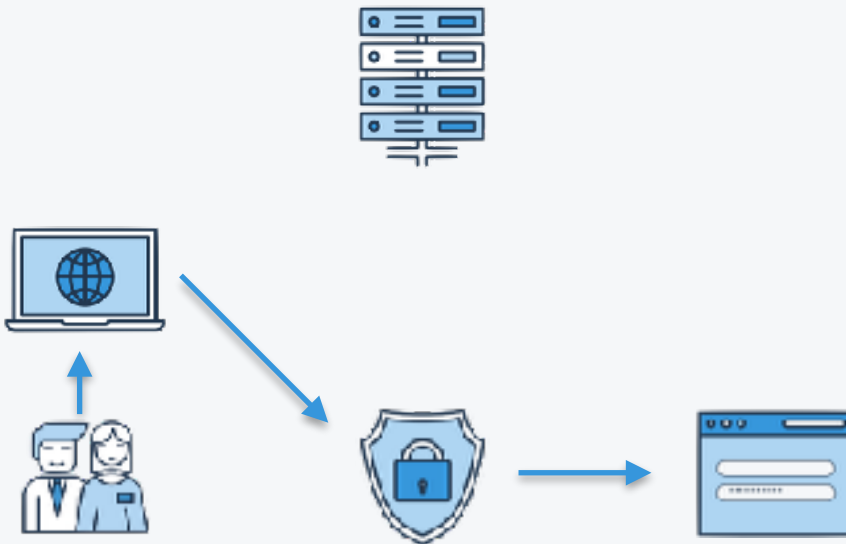
Authentication Flows

Implicit Flow



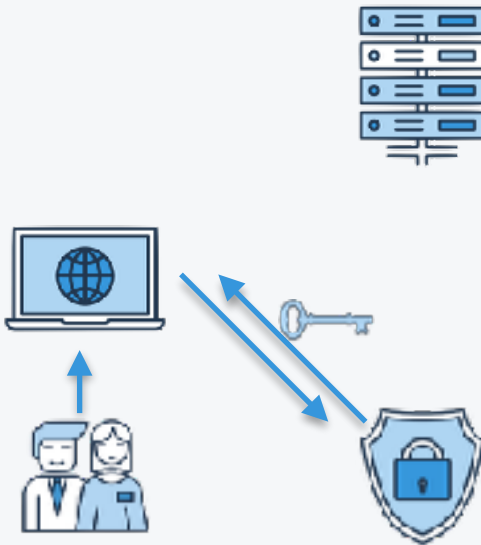
Authentication Flows

Implicit Flow



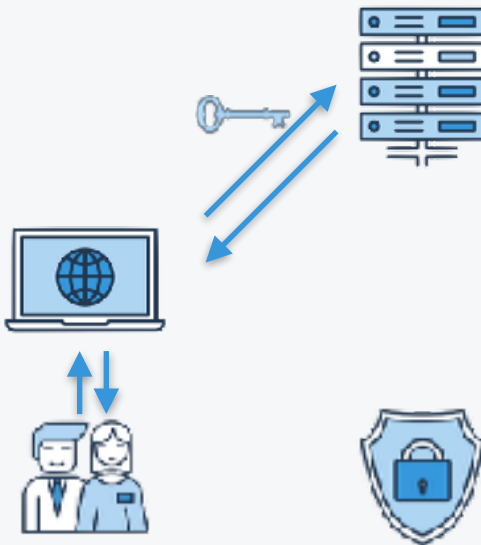
Authentication Flows

Implicit Flow



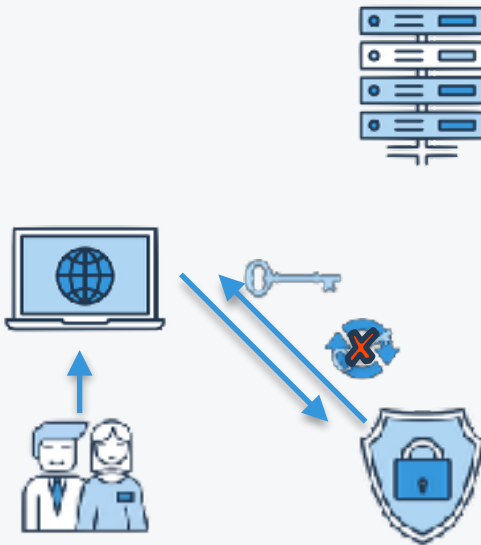
Authentication Flows

Implicit Flow



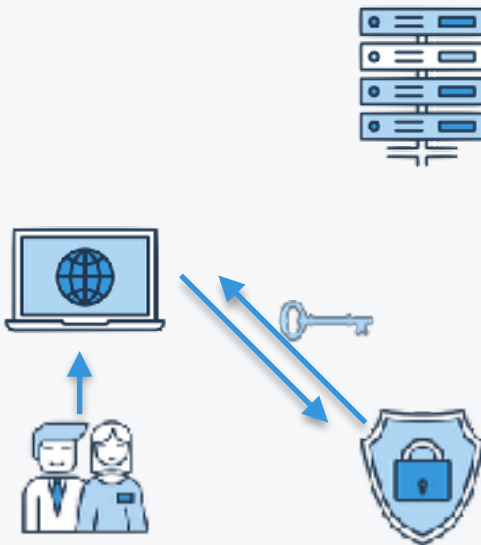
Authentication Flows

Implicit Flow



Authentication Flows

Implicit Flow



Tokens 101

OAuth

Tokens

Access Token



- Give you access to a resource
- Controls access to your API
- Short lived

Refresh Token



- Enables you to get a new token
- Long lived
- Can be revoked



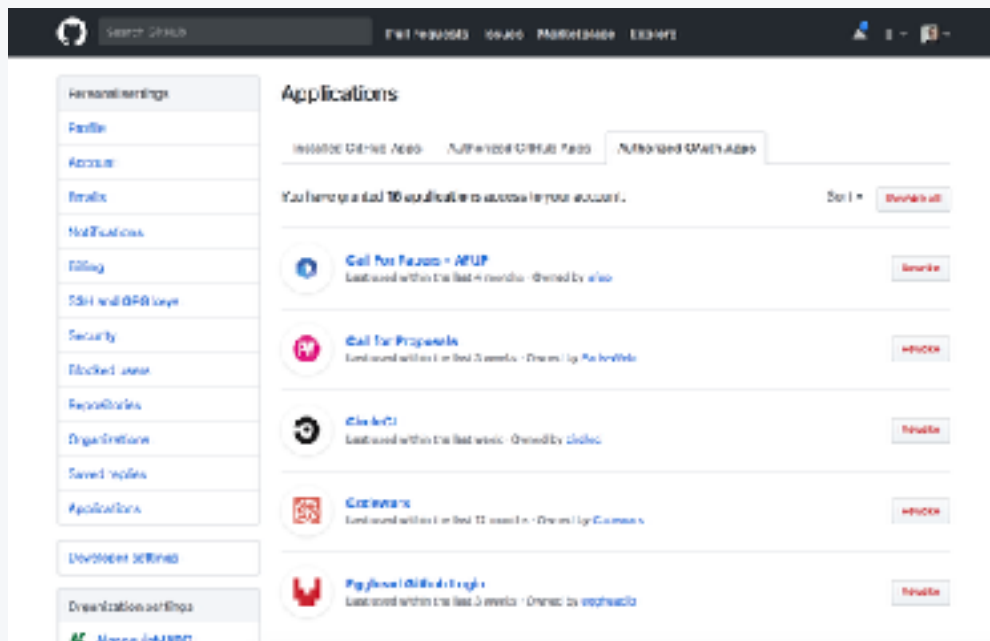
OAuth

Tokens

Refresh Token



- Enables you to get a new token
- Longed lived
- Can be revoked



@joel__lord
#iJS18

OAuth

Tokens

Refresh Token



- Enables you to get a new token
- Longed lived
- Can be revoked

← Applications ayant accès à votre compte		
	App in the Air - travel planner & flight tracker	Droits d'accès accédés : Gmail, Google Calendar, Google Contacts, informations de base relatives au compte
	Dev - Callaway Golf Account	Certains droits d'accès au compte accédés : Google+, informations de base relatives au compte
	Dev Menu Application	Certains droits d'accès au compte accédés : Google+, informations de base relatives au compte
	DOODLE	Certains droits d'accès au compte accédés : Google Drive, Google Play, Google+, informations de base relatives au compte
	Doodle	Droits d'accès accédés : Google Calendar, Google Contacts, informations de base relatives au compte
	draw.io	Droits d'accès accédés : Google Drive, informations de base relatives au compte
	Dropbox	Droits d'accès accédés : Google Contacts, informations de base relatives au compte
	Duolingo: Learn Languages	Certains droits d'accès au compte accédés : Google+

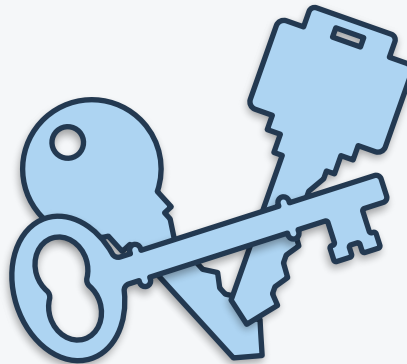


@joel__lord
#iJS18

OAuth

Tokens

- WS-Federated
- SAML
- JWT
- Custom stuff
- More...



JSON Web Token

- Header
- Payload
- Signature

Header

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

Payload

```
{  
  "sub": "1234567890",  
  "name": "Joel Lord",  
  "scope": "posts:read posts:write"  
}
```

Signature

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

JSON Web Token

- Header
- Payload
- Signature

Header

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9

Payload

eyJzdWliOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvcZWwgTG9yZCIsImFkbWlucj0cnVILCJzY29wZSI6InBvc3RzOnJiYWQgcG9zdHM6d3JpdGUifQ

Signature

XesR-pKdlscHfUwoKvHnACqfpe2ywJ6t1BJKsq9rEcg

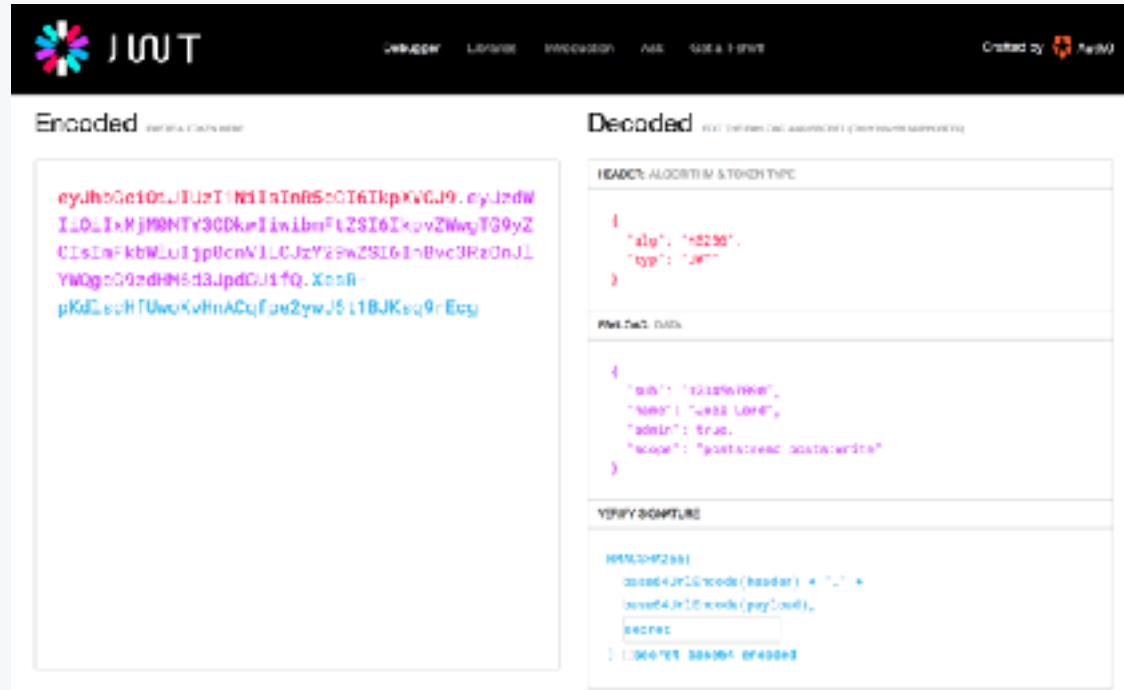
JSON Web Token

- Header
- Payload
- Signature

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvcZWwgTG9yZCIsImFkbWlucjpwcnVILCJzY29wZSI6InBvc3RzOnJlYWQgcG9zdHM6d3JpdGUifQ.XesR-  
pKdlscHfUwoKvHnACqfpe2ywJ6t1BJKsq9rEcg
```

JSON Web Token

- Header
- Payload
- Signature



The image shows the JWT.io web application interface. At the top, there's a navigation bar with the JWT logo, links for 'Debugger', 'Library', 'Introduction', 'API', and 'About', and a 'Created by' section with a GitHub icon. The main content area is split into two panels: 'Encoded' on the left and 'Decoded' on the right. The 'Encoded' panel contains a long string of base64-encoded characters. The 'Decoded' panel shows the decoded structure of the token, which is a JSON object with three parts: 'alg' (HS256), 'typ' (JWT), and a payload. The payload contains fields for 'sub' (123456789), 'name' (john doe), 'admin' (true), and 'scope' (posts:read, posts:write). Below the decoded JSON, there's a 'VERIFY SIGNATURE' section showing the algorithm (HS256) and the signature (a long base64 string).

Encoded

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWwIj0iIiwiaWF0Ij0iMTYzODk1IiwiaXNjaWkiOiJLSiIsImV4cCI6MTYzODk1LCJpbnNpY2kiOiJYMDp0Q9zdHMsIjpdGUiOiJq.Xasfi-pKdLschTUwvHnACqfpe2ywJ5l1BjKsq9rEcj
```

Decoded

HEADER: ALGORITHM & TOKEN TYPE

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

Payload: DATA

```
{  "sub": "123456789",  "name": "john doe",  "admin": true,  "scope": "posts:read posts:write"}
```

VERIFY SIGNATURE

```
HS256(123456789)  base64url_encode(header) + "." +  base64url_encode(payload),  SECRET  [3047e7 885b6a 8f688e3]
```

Image: <https://jwt.io>

Codiiiiing Time!

Auth Server

```
var express = require('express');
var Webtask = require('webtask-tools');
var bodyParser = require('body-parser');
var jwt = require("jsonwebtoken");
var app = express();

var users = [
  {id: 1, username: "joellord", password: "joellord"},
  {id: 2, username: "guest", password: "guest"}
];

app.use(bodyParser.urlencoded());

app.get("/login", function(req, res) {
  var loginForm = "<form method='post'><input type=hidden name=callback value='" +
    req.query.callback + "'><input type=text name=username /><input type=text name=password /><input type=submit></form>";
  res.status(200).send(loginForm);
});

app.post("/login", function(req, res) {
  if (!req.body.username || !req.body.password) return res.status(400).send("Need username and password");

  var user = users.find(function(u) {
    return u.username === req.body.username && u.password === req.body.password;
  });
  if (!user) return res.status(401).send("User not found");

  var token = jwt.sign({
    sub: user.id,
    scope: "api:read",
    username: user.username
  }, "mysupersecret", {expiresIn: "10 minutes"});

  res.redirect(req.body.callback + "#access_token=" + token);
});

app.get('*', function (req, res) {
  res.sendStatus(404);
});
```

API

```
var express = require('express');
var Webtask = require('webtask-tools');
var bodyParser = require('body-parser');
var randopeep = require("randopeep");
var jwt = require("jsonwebtoken");
var app = express();

var users = [
  {id: 1, username: "joellord", password: "joellord"},
  {id: 2, username: "guest", password: "guest"}
];

app.use(bodyParser.json());

app.post("/login", function(req, res) {
  if (!req.body.username || !req.body.password) return res.status(400).send("Need username and password");

  var user = users.find(function(u) {
    return u.username === req.body.username && u.password === req.body.password;
  });
  if (!user) return res.status(401).send("User not found");

  var token = jwt.sign({
    sub: user.id,
    scope: "api:read",
    username: user.username
  }, "mysupersecret", {expiresIn: "10 minutes"});

  res.status(200).send({token: token});
});

app.get('*', function (req, res) {
  res.sendStatus(404);
});

module.exports = Webtask.fromExpress(app);
```

Auth Server

```
var express = require('express');  
var bodyParser = require('body-parser');  
var jwt = require("jsonwebtoken");  
var app = express();  
  
// ...
```

Auth Server

```
var express = require('express');  
var bodyParser = require('body-parser');  
var jwt = require("jsonwebtoken");  
var app = express();  
  
// ...
```

Auth Server

```
var express = require('express');  
var bodyParser = require('body-parser');  
var jwt = require("jsonwebtoken");  
var app = express();  
  
// ...
```

Auth Server

```
var express = require('express');  
var bodyParser = require('body-parser');  
var jwt = require("jsonwebtoken");  
var app = express();
```

```
// ...
```

Auth Server

```
// Requires ...
```

```
var users = [  
  {id: 1, username: "joellord", password: "joellord"},  
  {id: 2, username: "guest", password: "guest"}  
];
```

Auth Server

```
// Requires ...
```

```
var users = [...];
```

```
app.use(bodyParser.urlencoded());
```

```
app.post("/login", function(req, res) {  
  // POST for login  
});
```

```
app.get('*', function (req, res) {  
  res.sendStatus(404);  
});
```

Auth Server

```
// Requires ...
```

```
var users = [...];
```

```
app.use(bodyParser.urlencoded());
```

```
app.post("/login", function(req, res) {  
  // POST for login  
});
```

```
app.get('*', function (req, res) {  
  res.sendStatus(404);  
});
```


Auth Server

```
app.post("/login", function(req, res) {  
  // POST for login  
  if (!req.body.username || !req.body.password)  
    return res.status(400).send("Need username and password");  
  
  var user = users.find(function(u) {  
    return u.username === req.body.username && u.password === req.body.password;  
  });  
  if (!user) return res.status(401).send("User not found");  
  
  var token = jwt.sign({  
    sub: user.id,  
    scope: "api:read",  
    username: user.username  
  }, "mysupersecret", {expiresIn: "10 minutes"});  
  
  res.redirect(req.body.callback + "#access_token=" + token);  
});
```

Auth Server

```
app.post("/login", function(req, res) {  
  // POST for login  
  if (!req.body.username || !req.body.password)  
    return res.status(400).send("Need username and password");
```

```
  var user = users.find(function(u) {  
    return u.username === req.body.username && u.password === req.body.password;  
  });  
  if (!user) return res.status(401).send("User not found");
```

```
  var token = jwt.sign({  
    sub: user.id,  
    scope: "api:read",  
    username: user.username  
  }, "mysupersecret", {expiresIn: "10 minutes"});  
  
  res.redirect(req.body.callback + "#access_token=" + token);  
});
```

Auth Server

```
app.post("/login", function(req, res) {  
  // POST for login  
  if (!req.body.username || !req.body.password)  
    return res.status(400).send("Need username and password");  
  
  var user = users.find(function(u) {  
    return u.username === req.body.username && u.password === req.body.password;  
  });  
  if (!user) return res.status(401).send("User not found");  
  
  var token = jwt.sign({  
    sub: user.id,  
    scope: "api:read",  
    username: user.username  
  }, "mysupersecret", {expiresIn: "10 minutes"});  
  
  res.redirect(req.body.callback + "#access_token=" + token);  
});
```

Auth Server

```
app.post("/login", function(req, res) {  
  // POST for login  
  if (!req.body.username || !req.body.password)  
    return res.status(400).send("Need username and password");  
  
  var user = users.find(function(u) {  
    return u.username === req.body.username && u.password === req.body.password;  
  });  
  if (!user) return res.status(401).send("User not found");  
  
  var token = jwt.sign({  
    sub: user.id,  
    scope: "api:read",  
    username: user.username  
  }, "mysupersecret", {expiresIn: "10 minutes"});  
  
  res.redirect(req.body.callback + "#access_token=" + token);  
});
```

Auth Server

```
// Requires ...
```

```
var users = [...];
```

```
app.use(bodyParser.urlencoded());
```

```
app.post("/login", function(req, res) {  
  // POST for login  
});
```

```
app.get('*', function (req, res) {  
  res.sendStatus(404);  
});
```

```
app.listen(8080, () => console.log("Auth server running on 8080"));}
```

API

```
var express    = require('express');  
var bodyParser = require('body-parser');  
var randopeep = require("randopeep");  
var expressjwt = require("express-jwt");  
var app = express();
```

API

```
var express    = require('express');  
var bodyParser = require('body-parser');  
var randopeep = require("randopeep");  
var expressjwt = require("express-jwt");  
var app = express();
```

API

```
var express    = require('express');  
var bodyParser = require('body-parser');  
var randopeep = require("randopeep");  
var expressjwt = require("express-jwt");  
var app = express();
```


API

```
var express    = require('express');  
var bodyParser = require('body-parser');  
var randopeep  = require("randopeep");  
var expressjwt = require("express-jwt");  
var app = express();
```

API

```
var express      = require('express');  
var bodyParser   = require('body-parser');  
var randopeep    = require("randopeep");  
var expressjwt   = require("express-jwt");  
var app = express();
```

API

// Requires ...

```
var jwtCheck = expressjwt({  
  secret: "mysupersecret"  
});
```

API

```
// Requires and config ...
```

```
app.get("/headline", function(req, res) {  
  // Unprotected  
  res.status(200).send(randopeep.clickbait.headline());  
});
```

```
app.get("/protected/headline", jwtCheck, function(req, res) {  
  // Protected  
  res.status(200).send(randopeep.clickbait.headline("Joel Lord"));  
});
```

```
app.get('*', function (req, res) {  
  res.sendStatus(404);  
});
```

API

```
// Requires and config ...
```

```
app.get("/headline", function(req, res) {  
  // Unprotected  
  res.status(200).send(randopeep.clickbait.headline());  
});
```

```
app.get("/protected/headline", jwtCheck, function(req, res) {  
  // Protected  
  res.status(200).send(randopeep.clickbait.headline("Joel Lord"));  
});
```

```
app.get('*', function (req, res) {  
  res.sendStatus(404);  
});
```

API

```
// Requires and config ...
```

```
app.get("/headline", function(req, res) {  
  // Unprotected  
  res.status(200).send(randopeep.clickbait.headline());  
});
```

```
app.get("/protected/headline", jwtCheck, function(req, res) {  
  // Protected  
  res.status(200).send(randopeep.clickbait.headline("Joel Lord"));  
});
```

```
app.get('*', function (req, res) {  
  res.sendStatus(404);  
});
```

API

```
// Requires and config ...
```

```
app.get("/headline", function(req, res) {  
  // Unprotected  
  res.status(200).send(randopeep.clickbait.headline());  
});  
  
app.get("/protected/headline", jwtCheck, function(req, res) {  
  // Protected  
  res.status(200).send(randopeep.clickbait.headline("Joel Lord"));  
});
```

```
app.get('*', function (req, res) {  
  res.sendStatus(404);  
});
```

API

```
// Requires and config ...
```

```
app.get("/headline", function(req, res) {  
  // Unprotected  
});
```

```
app.get("/protected/headline", jwtCheck, function(req, res) {  
  // Protected  
});
```

```
app.get('*', function (req, res) {  
  res.sendStatus(404);  
});
```

```
app.listen(8888, () => console.log("API listening on 8888"));
```


Add the headers

Request URL: <https://wl-13aeb74eaaa99135c2725d4a98edd49e-0.run.webtask.io/clickbaiter/protected/headline>

Request Method: GET

Status Code: 200 OK

Remote Address: 54.183.32.187:443

Referrer Policy: no-referrer-when-downgrade

► Response Headers (130)

[View source](#)

Accept 4/4

Accept-Encoding: gzip, deflate, br

[illegible][illegible]

Cache-Control: no-cache

Connection: keep-alive

Host: wt-13aebf4a0aa9913547725e4a98a4d49c-8.run.webtask.io

Origine: <http://www.burton.com>

Pragma: no-cache

Referans: <http://localhost:5080/>

User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_13_3) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/65.0.3325.162 Safari/537.36



@joel__lord
#iJS18

Live Demo

[https://github.com/joellord/
secure-spa-auth0](https://github.com/joellord/secure-spa-auth0)



A close-up, low-angle shot of a hockey stick's blade and a black puck resting on a light blue ice surface. The ice is marked with numerous scratches and scuffs. The lighting is dramatic, coming from the upper left, casting a soft shadow of the stick and puck onto the ice. The stick is positioned diagonally across the upper right portion of the frame.

Delegation!









CANADA



PASSPORT
PASSEPORT





Human Resources
Development Canada

Développement des
ressources humaines Canada

SOCIAL
INSURANCE
NUMBER

NUMÉRO
D'ASSURANCE
SOCIALE

123 123 123

PETAR PETROVIC

Front



- Driver's licence number (4a)
- Last name (1)
- First name(s) (2)
- Date of birth (3)
- Cardholder's address (4)
- Driver's licence class(es) (6)*
- Sex (5)
- Condition(s) (7)*
- Height (cm) (8)
- Eye colour (10)
- Endorsement(s) (9a)*
- Reference number (5)
- Date of expiry (12b)
- Date of issue (12a)

Back



Two-dimensional bar code

The two-dimensional bar code (2D) contains all licence information except the photo and signature.

Definitions of the licence classes, conditions and endorsements listed on the front

Bar code

The bar code corresponds to the driver's licence number.

Telephone number for more information

Telephone number to check the validity of a licence (taxis apply)

Introducing OpenID Connect

OpenID Connect

- Built on top of OAuth 2.0
- OpenID Connect (OIDC) is to OpenID what Javascript is to Java
- Provides Identity Tokens in JWT format
- Uses a /userinfo endpoint to provide the info



OpenID Connect

Scopes

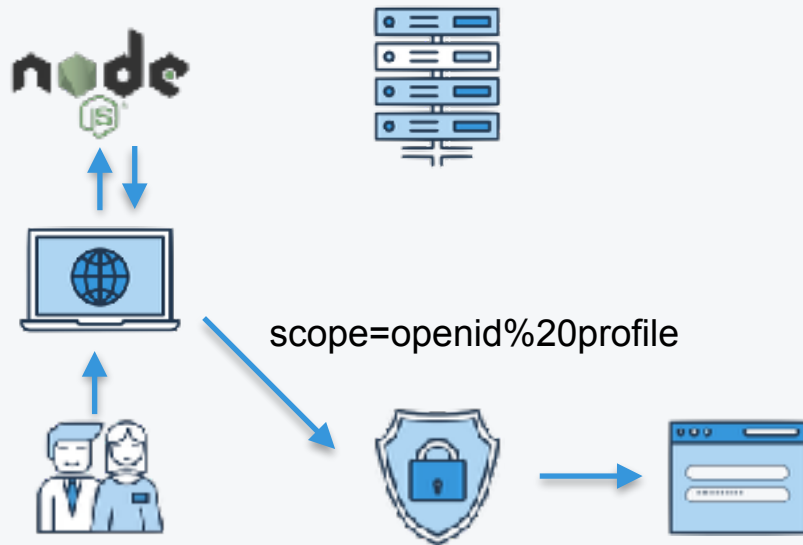
- openid
- profile
- email
- address
- phone



@joel__lord
#iJS18

OpenID Connect Flows

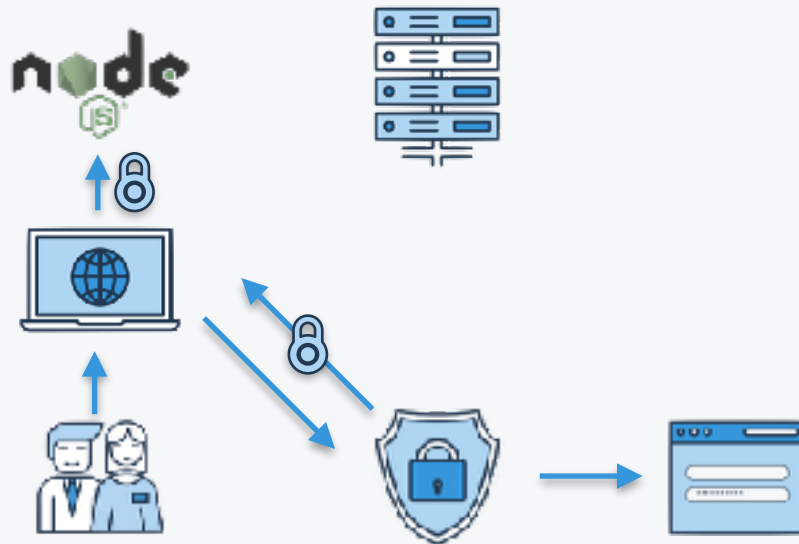
Authorization Code



@joel__lord
#iJS18

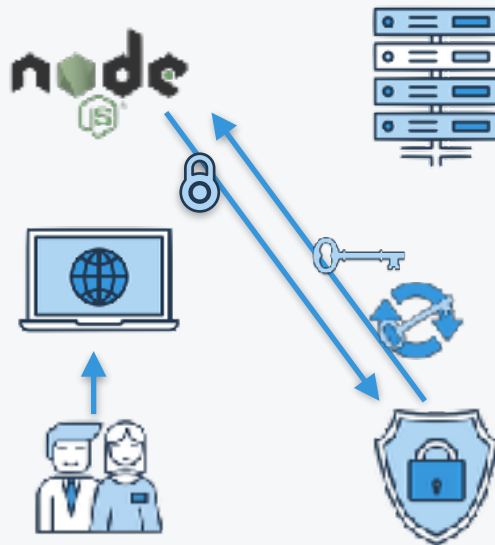
Authentication Flows

Authorization Code



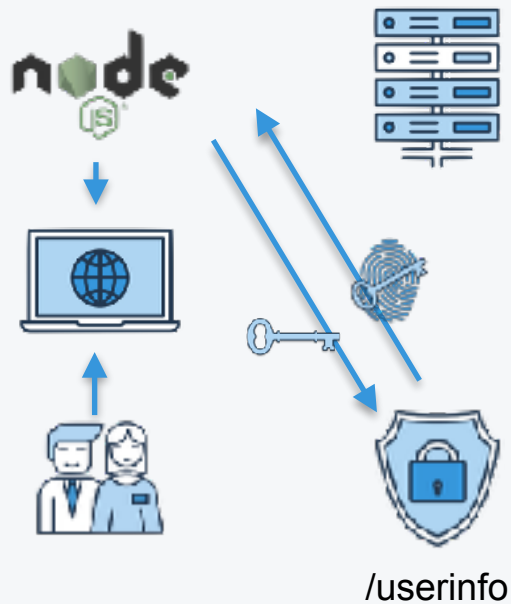
Authentication Flows

Authorization Code



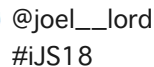
Authentication Flows

Authorization Code



Full flow

The screenshot shows the top section of the OpenID Connect Playground website. At the top left is the OpenID logo. To its right are navigation links: "Debugger", "Introduction", "FAQ", and "Contact Us". On the far right, it says "Powered by Auth0". The main heading is "OpenID Connect Playground" in orange. Below it, a paragraph states: "The OICP playground is for developers to test and work with OpenID Connect easily, by simplifying them more insight into how OpenID Connect works." Below this is a horizontal bar with three items: "Main", "OPENID CONNECT + CRYPTO" (which is highlighted with a white background), and "Debugger". To the right of this bar is a button labeled "START DEMO HERE". Below the bar, there's a numbered step indicator "1" in a circle, followed by the text "Redirect to OpenID Connect Server". Below this text is a dark rectangular area representing a terminal or log output, showing several lines of JSON-like data related to an authentication request.



A close-up, low-angle shot of a hockey stick's blade and a puck resting on a textured ice surface. The stick is dark and worn, with the blade pointing towards the left. The puck is black and cylindrical, positioned just in front of the stick's blade. The ice is light blue-grey with numerous fine scratches and scuffs. A dark, semi-transparent rectangular bar is overlaid horizontally across the middle of the image, containing the word "Delegation!" in white, bold, sans-serif font.

Delegation!



I Don't Care About Security

iJS, London, UK

April 11, 2018



@joel__lord



joellord