It is quite similar to Authorization grand flow, the only difference is that **Implicit grant flow** does not send the auth back to the client application.

The user logins to some web application (which is client application) and tries to use or get some resources (for example upload a photo). The client application gives an option to upload the photos from google photos application, and that option the user chooses. Then the client application sends the request to the authorization server on behalf of the user. In that request client application sends very important information to the authorization server. It is a **scope** - it means the type of access that is required by client application on behalf of the user. In that example it might ask for access to view and download the photos from google photos app. In other words, a client application with the scope will inform the authorization server what it needs. Then authorization server requests for authorization from a user (username : password), after the authorization is succeeded server asks the user to provide permission for the client application, while doing so the server will show to user what exactly the client application wants to do (kind of permission that is requested). And also the authorization server will ask the user whether the user trusts the client application. If the user approves, then authorization server sends to the client application **access token** (not auth token), and there is no backend server involved.

Access token is short time lived token and it will have expiry time (15 -30min). If it is refresh token it remains valid until the user is logged in on the browser. Once the access token is expired the client application can use refresh token, to renew the access token as long as the refresh token is valid.

If the user logged out of google photo application, all the process will be repeated. Once the backend server gets the access token from an authorized server it can send a token to google photo application to request for the photo (resource). And the backend server will send the token as part of the API call. The google photo app. will validate the access token if it is valid it will send back requested resources (in our example - photos). So with this the client application has got access to the protected resources that are the photo. And it gets access on behalf of the resource owner, because the user (resource owner) authorized the client application to access the photos.

NOTICE: all that flow happening on a front end side (channel), that is browser. There is no backend chanel, this is not secure, because the access token is available in the browser, and can be easily stolen. The only security is involved that is the access token short time lived. Typically this flow is used for JavaScript applications that runs inside the browser. The application that does not have backend server.

The example of implemented Implicit flow, done by google:

<https://developers.google.com/identity/protocols/oauth2/javascript-implicit-flow>