

DSA001 DMeta SuperApp - Login

Figure 1: Initial login screen on DMeta SuperApp running on Android device, viewed using scrpcy and React Native running in background terminal.

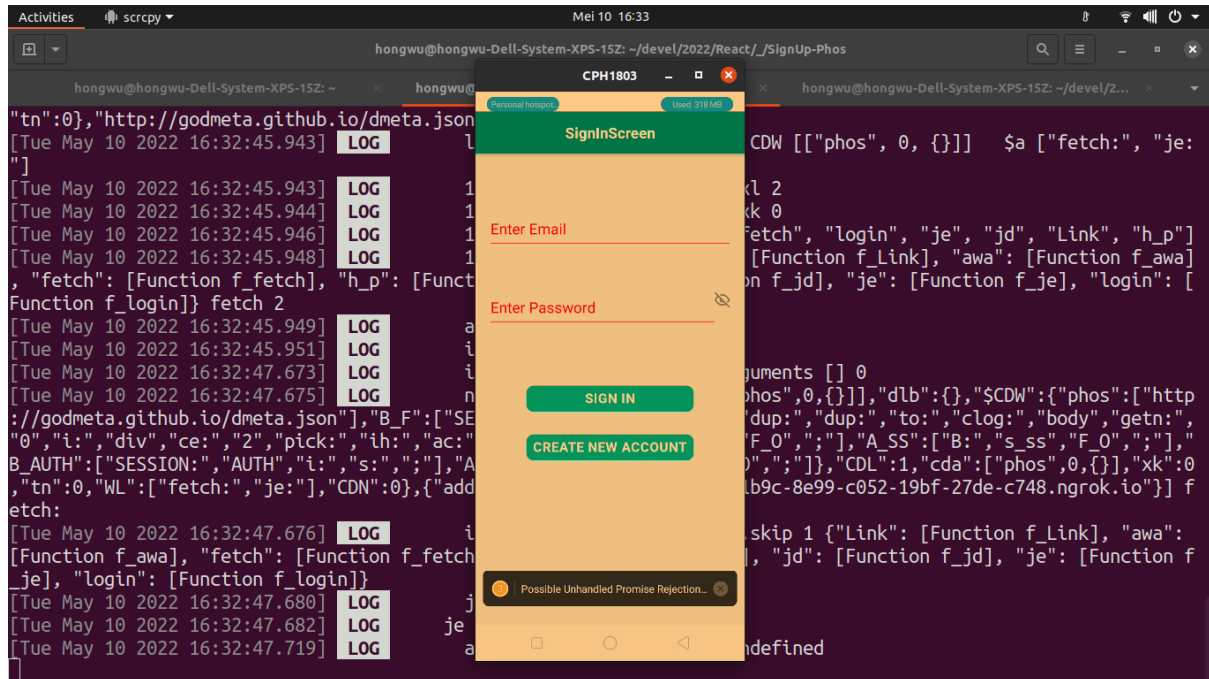


Figure 2: Login successfully.

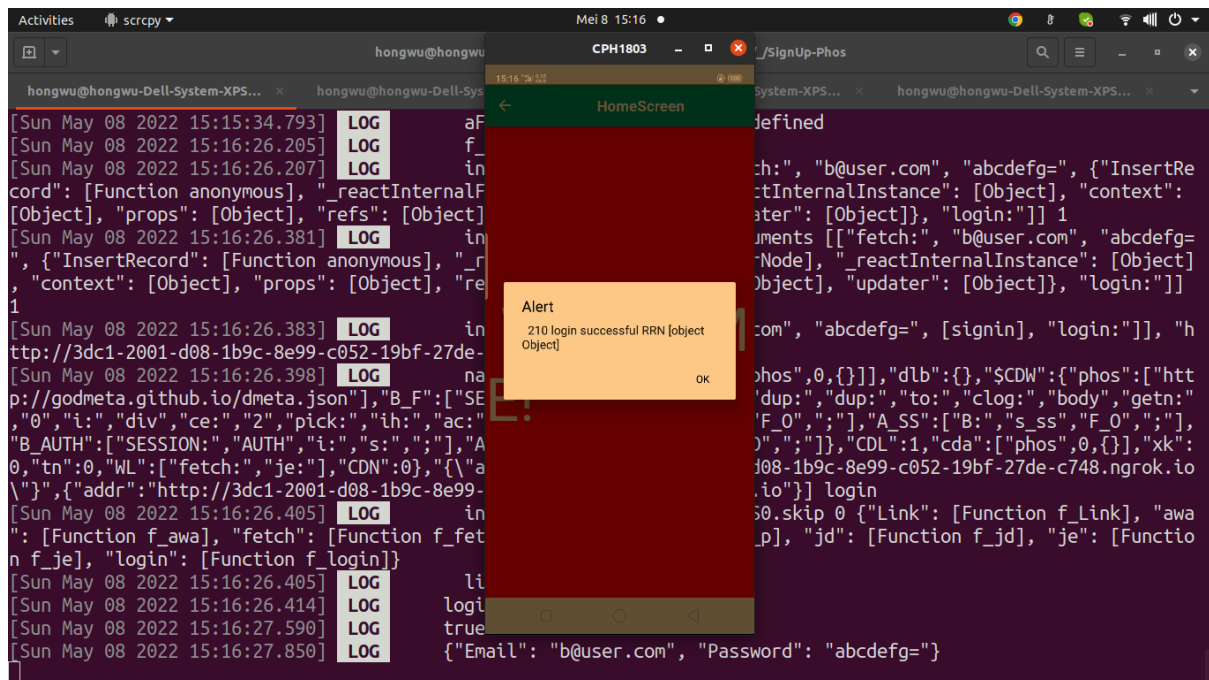
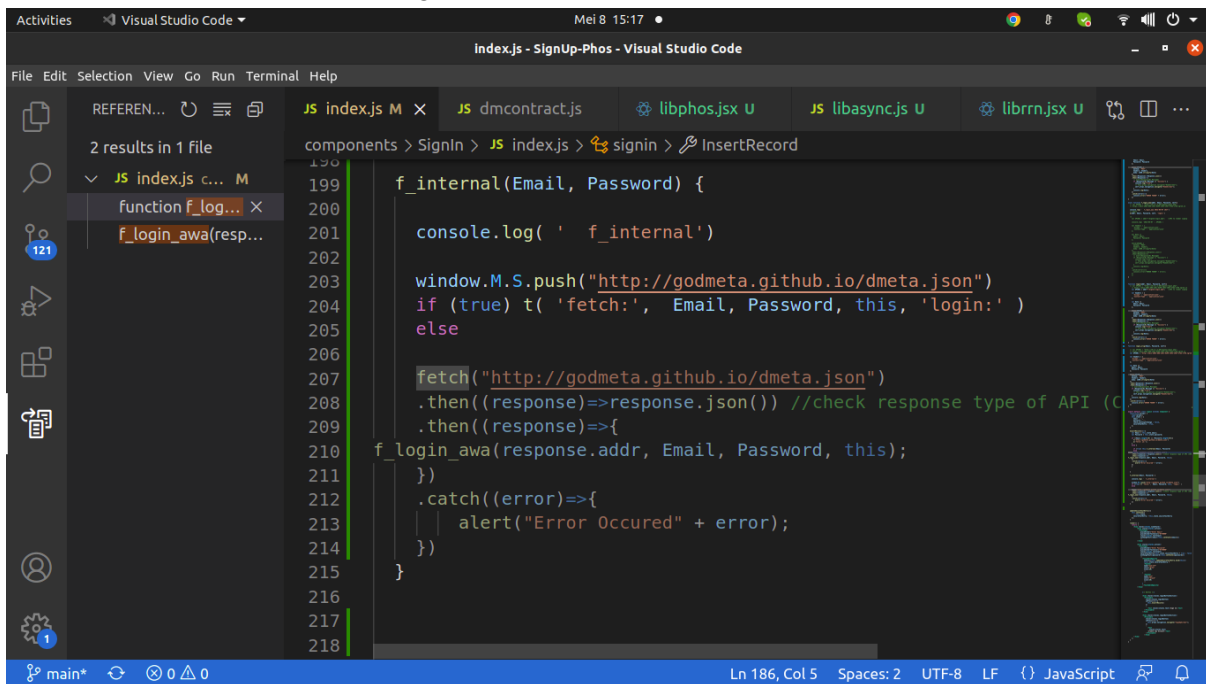


Figure 3: Proscription Reverse React Notation one liner (line 204)
vs. original React code (lines 207 to 214)



DMeta SuperApp: Login

DMeta SuperApp is the name of the mobile plus web architecture and application created based on DMeta server and Phoscript, a metaprogramming script derived from FORTH programming language, that aims to achieve the following:

- Simplifying mobile and web apps programming, from lengthy blocky codes that typically span over 10 lines to FORTH or Phoscript one liners.
- Providing a platform where programmers can collaborate and CO-OWN source code, avoiding the need of rewriting an app from scratch, so that they may build a SuperApp together literally ONE LINE AT A TIME using Phoscript.
- Programmers use hash to identify code at function level, gaining greater control while negotiating with investors.
- Hash of Phoscript Code (HPC) allows portability of algorithms across multiple programming languages.

In short, DMeta SuperApp could be the biggest breakthrough in free software or open source ecosystems since GCC and Linux, allowing programmers to own and control the use of source code DOWN TO ONE LINE OF CODE.

In this tutorial, we show (figure 3) how React source code (lines 207 to 214) can be simplified to one single line of Phoscript Reverse React Notation (line 204).

The original source repository can be found in:

- [DAR001 DMeta Mobile App using Reverse React Notation – Tunnel Redirection](#)
- <https://docs.google.com/document/d/13OraMz9NLU4Uf9-Kso-OHbAwbzdJq3S-lse0r98N1u4/edit?usp=drivesdk>



The Tower of Babel and the Language of Light – Phoscript

If your grandfather is still alive, would you ask if his grandfather understood what the atomic bomb, television and mobile phones were, before they were invented?

In fact, legends have it that they were recorded in ancient sacred texts in the Orient.

If we tell you that a universal language that is capable of uniting all humankind has been invented in the 1960s, would you believe it?

This time, the legend of all humankind united by one language was recorded in ancient sacred texts not in the Orient, but the near East, yet, written in the reverse order.

With this language of Light, or Phoscript, derived from FORTH and reverse polish notation, humankind may reverse the effect of the Tower of Babel.

And as they said, the rest is history, only this time, YOU will be writing it

https://en.m.wikipedia.org/wiki/Tower_of_Babel

- The Tower of Babel (Hebrew: מִגְדַּל בָּבֶל, Migdal Bavel) narrative in Genesis 11:1–9 is an origin myth meant to explain why the world's peoples speak different languages.[1][2][3][4]

- According to the story, a united human race in the generations following the Great Flood, speaking a single language and migrating eastward, comes to the land of Shinar (שִׁנְאָר). There they agree to build a city and a tower tall enough to reach heaven. God, observing their city and tower, confounds their speech so that they can no longer understand each other, and scatters them around the world.

React / Javascript functions related to Phoscript Reverse React Notation

(full source code is available upon request)

```
f_internal(Email, Password) {
  window.M.S.push("http://godmeta.github.io/dmeta.json")
  if (true) t( 'fetch:', Email, Password, this, 'login:' )
  else
    fetch("http://godmeta.github.io/dmeta.json")
    .then((response)=>response.json()) //check response type of API
(CHECK OUTPUT OF DATA IS IN JSON)
    .then((response)=>{
      f_login_awa(response.addr, Email, Password, this);
    })
    .catch((error)=>{
      alert("Error Occured" + error);
    })
}
```

```
t( 'fetch:', Email, Password, this, 'login:' )
```

```
async function t() { // call async functions with then
  var $l = strlen(arguments[0]);
  var $fn = substr(arguments[0], 0, $l - 1);
  if (in_array($fn, array_keys(M.AF))) {
    M.AF[$fn](arguments)
  }
}
```

```

async function f_fetch() {
    var S0 = window.M.S[0]
    var M = window.M
    await fetch(window.M.S.pop())
    .then((response)=>response.json()) //check response type of API
(CHECK OUTPUT OF DATA IS IN JSON)
    .then((response)=>{
        var S = window.M.S;
        window.M.S.push(response)
        if (arguments.length==0) n()
        else na(arguments, response.addr)
        return response;
    })
    .catch((error)=>{ alert("f_fetch Error Occured: " + error);})
}

```

```

// na next word in then function with arguments
function na() {
    var S0 = window.M.S[0]
    var M = window.M
    var S = window.M.S;

    var A=arguments;
    var c=arguments[0][0].length;
    var l=arguments[0][0][c-1].length;
    var $fn=arguments[0][0][c-1].substring(0,l-1);

    S.push(A[A.length-1])
    S.push(A[0][0][1])
    S.push(A[0][0][2])
    S.push(A[0][0][3])

    if (in_array($fn, array_keys(M.CF))) {
        M.CF[$fn] ();
    }
    return S[S.length-1];
}

```

```
async function f_login(){
  var S = window.M.S;
  var self=S.pop()
  var Password=S.pop()
  var Email=S.pop()
  var addr=S.pop()
  var APIURL = addr+"/SignIn/login.php";    //API to render signup
  var headers = {
    'Accept' : 'application/json',
    'Content-Type' : 'application/json'
  };
  var Data ={
    Email: Email,
    Password: Password
  };
  // MUST have await!!
  await fetch(APIURL,{
    method: 'POST',
    headers: headers,
    body: JSON.stringify(Data)
  })
  .then((Response)=>Response.json())
  .then((Response)=>{
    if (Response[0].Message == "Success") {
      self.props.navigation.navigate("HomeScreen");
    }
    console.log(Data);
  })
  .catch((error)=>{
    console.error("ERROR FOUND" + error);
  })
}
```