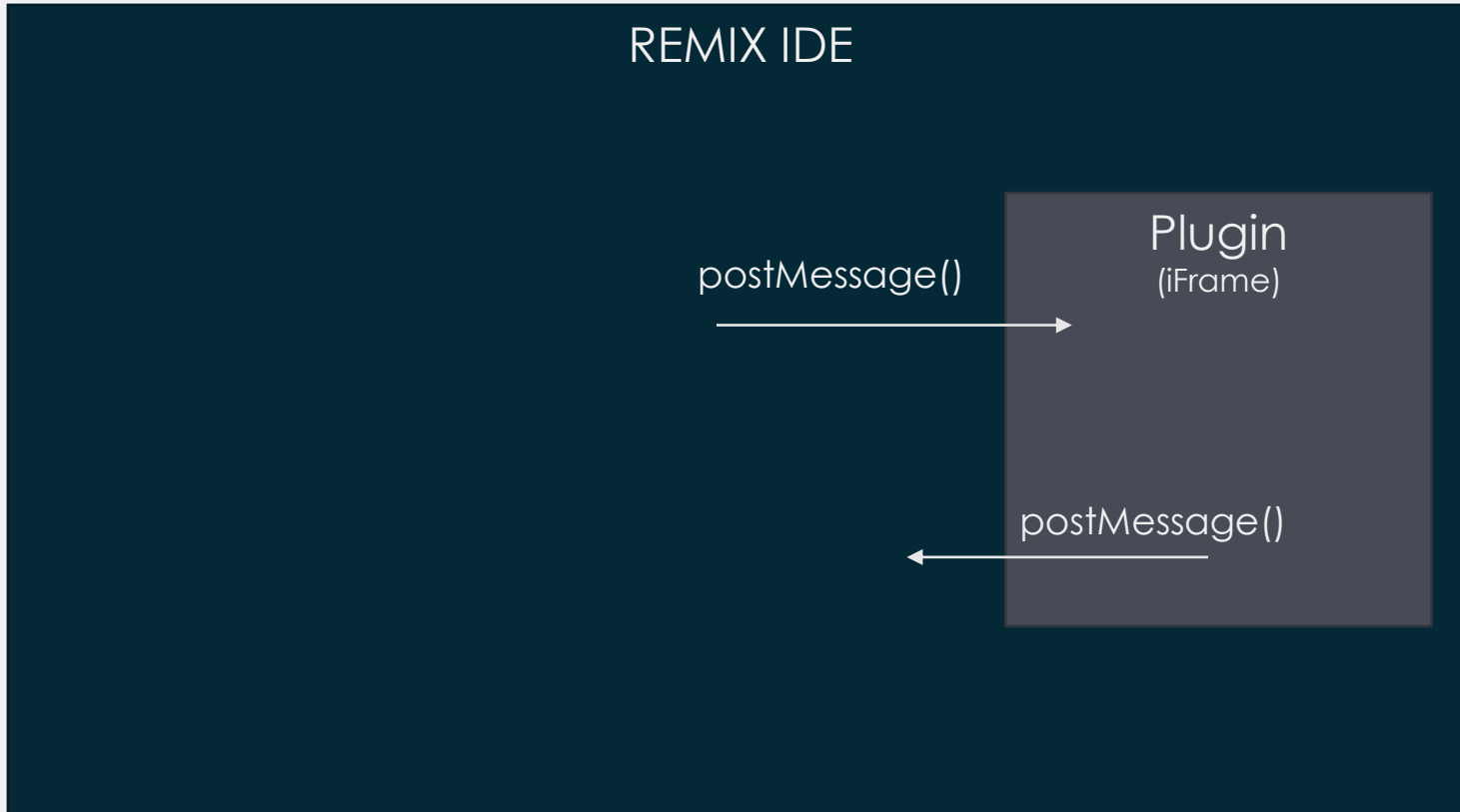


BUILD A PLUGIN FOR REMIX

Simple Plugin Hosted With Swarm

What is a Plugin ?



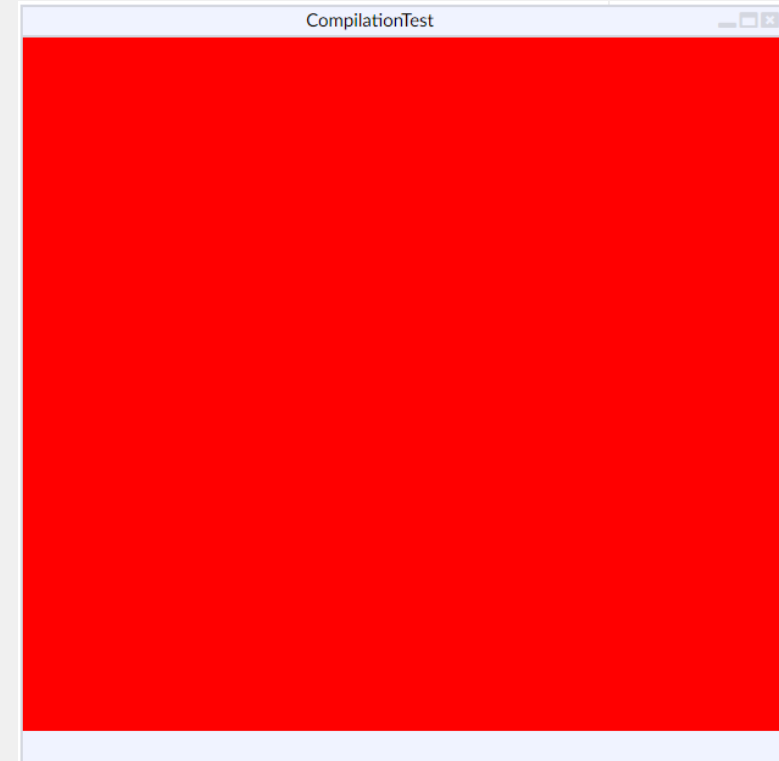
A Plugin is an Iframe loaded by Remix.

Communication is made with the [PostMessage API](#)

Our Goal



Compilation Succeed !



Compilation Failed !

Build and deploy a simple plugin that show green when compilation succeed, and red when compilation failed.

Index.html

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8" />
  <title>Simple Plugin</title>
</head>
<body>
</body>
<script>

</script>
</html>
```

Create a simple index.html file

Listen for messages

```
<script>
  const trustedOrigins = [
    "http://remix-alpha.ethereum.org",
    "http://remix.ethereum.org",
    "https://remix-alpha.ethereum.org",
    "https://remix.ethereum.org"
  ];
  // Listen to messages from Remix
  window.addEventListener('message', event => {
    // Only accept trusted origins
    if (trustedOrigins.indexOf(event.origin) === -1) {
      return;
    }
    const data = JSON.parse(event.data);
  }, false);
</script>
```

For security reason, we only accept trusted origins.
Then we parse the data from the message event.

Discover Remix Plugin API

The « event.data » object has 5 keys :

Key	Description
Action	can be a “notification”, “response” or “request”.
Key	where the message come from, or go to, in Remix.
Type	the type of message send.
Value	the content of the message (as a string).
id	the index of the message.

Discover Remix Plugin API (some examples)

Action	Key	Type
notification	compiler	compilationFinished
	txlistener	newTransaction
requests	config	getConfig
		setConfig
	compiler	getCompilationResult
	editor	getCurrentFile
		setFile
response	Same than requests	

Check all combinations here

Listen to compilationFinished

```
// Listen to messages from Remix
window.addEventListener('message', event => {
  // Only accept trusted origins
  if (trustedOrigins.indexOf(event.origin) === -1) {
    return;
  }
  const { type, value } = JSON.parse(event.data);
  // Listen to new compilation notification
  if (type === 'compilationFinished') {
    const success = value[0];
    document.body.style.backgroundColor = success ? 'green' : 'red';
  }
}, false);
```

The value of a compilationFinished message is an array with 2 entries.
The first one is a boolean which is true if compilation succeed

Note: compilationFinished is a « notification » action of key « compiler ».

Add some style

```
<style>
  html, body {
    height: 100%;
    margin: 0;
  }
</style>
```

Just add some css into you index.html

Deploy on Swarm



Download the [swarm binary](#) next to your index.html file.



Open your console and run :

```
swarm --bzzapi https://30400.swarm-gateways.net up index.html
```

It will return the hash of your file :

```
06f60ed9f1b5ac51c5ee2c4de1473904989b7cd66ab1bebed781cc1e1042f316
```

Use <https://30400.swarm-gateways.net> because it allows the result to be loaded inside an iframe.

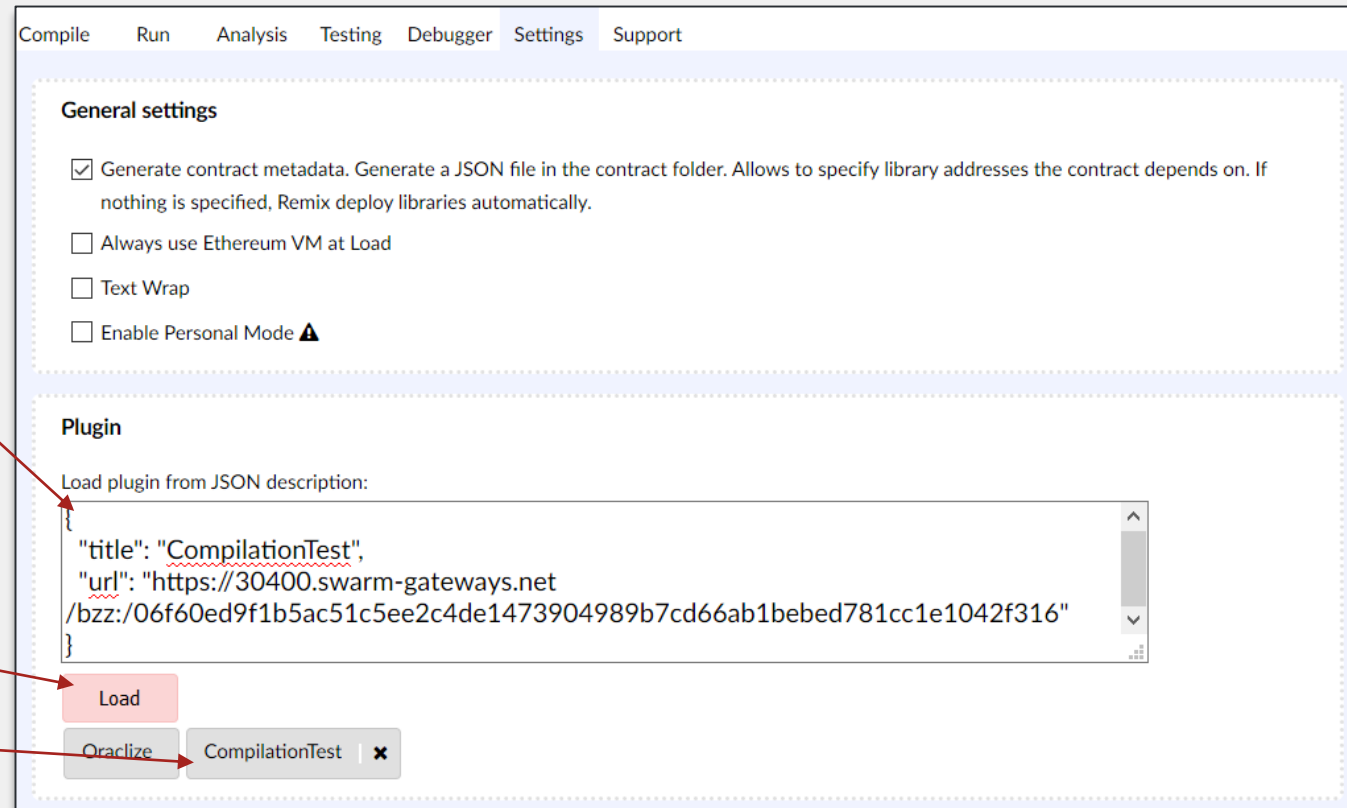
Load on Remix

◇ Go to [Remix](#), under settings > plugin add :

```
{  
  "title": "CompilationTest",  
  "url": "https://30400.swarm-  
gateways.net/bzz:/06f60ed9f1b5ac51c5e  
e2c4de1473904989b7cd66ab1bebed781cc1e  
1042f316"  
}
```

◇ Click « Load »

◇ And open your plugin



Compile a contract (succeed)

The screenshot displays the Solidity compiler interface. On the left, the source code for a contract named 'Test' is shown in a file named 'browser/Test.sol'. The code is as follows:

```
1 pragma solidity ^0.4.25;
2
3 contract Test {
4     string public name;
5
6     constructor() public {
7         name = "Compilation succeed";
8     }
9 }
10
```

On the right side of the interface, the 'Compile' tab is active. It shows the current version as '0.4.25+commit.59dbf8f1.Emscripten.clang'. Below this, there are checkboxes for 'Auto compile', 'Enable Optimization', and 'Hide warnings'. A 'Start to compile' button is present. The 'Test' dropdown menu is set to 'Test', and there are buttons for 'Details', 'ABI', and 'Bytecode'. A warning message states: 'Static Analysis raised 1 warning(s) that requires your attention. Click here to show the warning(s)'. Below this, a green box labeled 'Test' indicates the compilation was successful.

Compilation Succeed !

Compile a contract (failed)

The screenshot displays a web-based Solidity compiler interface. On the left, a code editor shows a Solidity contract named 'Test' with a constructor that sets 'name' to 'Compilation failed'. The code is as follows:

```
1 pragma solidity ^0.4.25;
2
3 contract Test {
4     string public name;
5
6     constructor() public {
7         name = "Compilation failed";
8     }
9 }
10
```

On the right, the 'Compile' tab is active. It shows the current version as '0.4.25+commit.59dbf8f1.Emscripten.clang'. There are checkboxes for 'Auto compile', 'Enable Optimization', and 'Hide warnings'. A 'Start to compile' button is present. Below this, there is a 'Swarm' button and buttons for 'Details', 'ABI', and 'Bytecode'. At the bottom, a red error message box indicates a 'ParserError: Expected ';' but got ''' at line 9, column 5.

CompilationTest

browser/Test.sol:9:5: ParserError: Expected ';' but got ''

```
}
^
```

Compilation Failed !