WebAssembly - Last

Application of WASM to WebApp

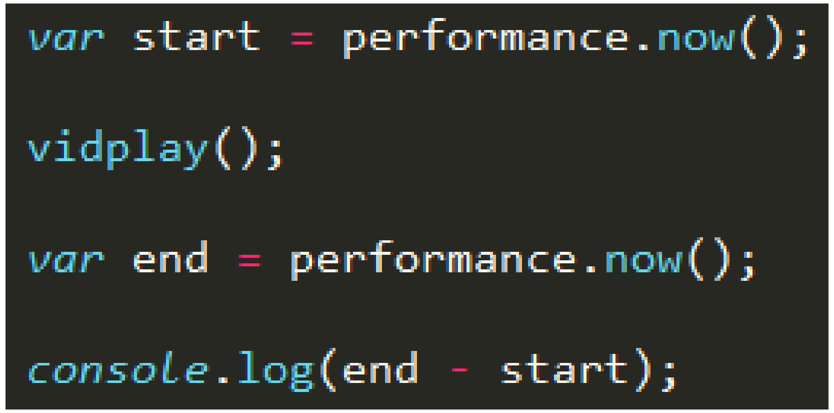
Content

- Memory / Table
- Application of WASM
- What have I learnt
- Todo

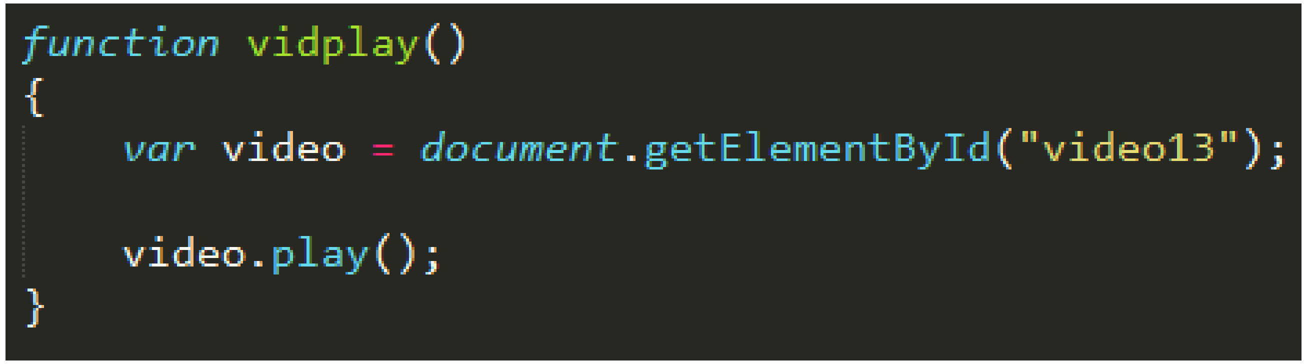
Memory / Table

	Memory	Table
Similarity	Array-like structure Accessible and mutable from both JS and WASM	
Difference	Resizable ArrayBuffer Holds raw bytes of memory	Given size and element type Stores function references

Application of WASM

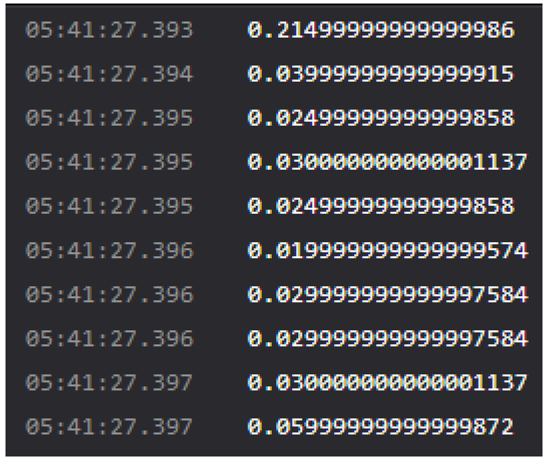


```
var start = performance.now();

vidplay();

var end = performance.now();

console.log(end - start);
```

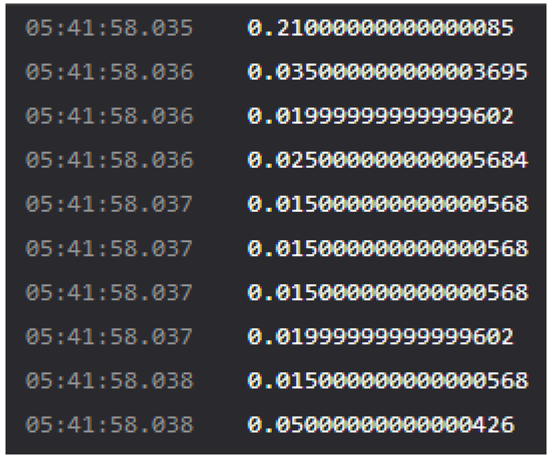


```
function vidplay()
{
    var video = document.getElementById("video13");

    video.play();
}
```

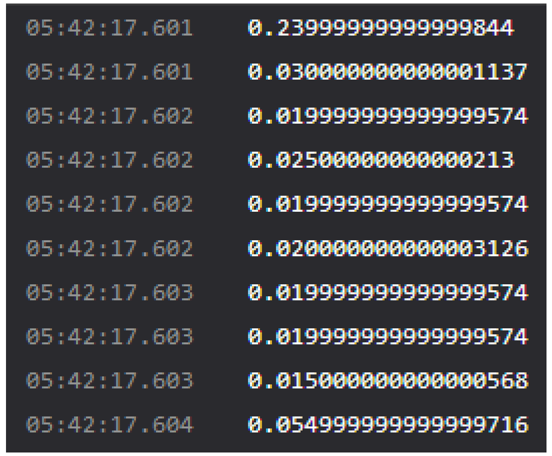
Application of WASM



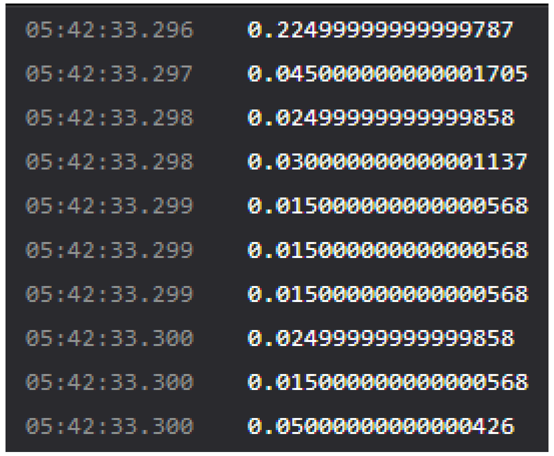
```
05:41:27.393    0.21499999999999986
05:41:27.394    0.03999999999999915
05:41:27.395    0.02499999999999858
05:41:27.395    0.030000000000001137
05:41:27.395    0.02499999999999858
05:41:27.396    0.019999999999999574
05:41:27.396    0.029999999999997584
05:41:27.396    0.029999999999997584
05:41:27.397    0.030000000000001137
05:41:27.397    0.05999999999999872
```



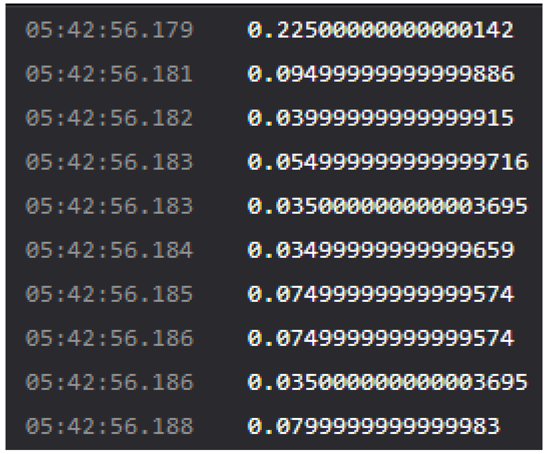
```
05:41:58.035    0.2100000000000085
05:41:58.036    0.035000000000003695
05:41:58.036    0.01999999999999602
05:41:58.036    0.025000000000005684
05:41:58.037    0.015000000000000568
05:41:58.037    0.015000000000000568
05:41:58.037    0.015000000000000568
05:41:58.037    0.01999999999999602
05:41:58.038    0.015000000000000568
05:41:58.038    0.05000000000000426
```



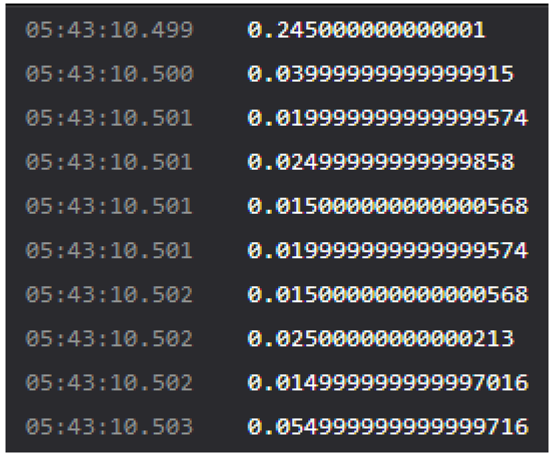
```
05:42:17.601    0.23999999999999844
05:42:17.601    0.030000000000001137
05:42:17.602    0.019999999999999574
05:42:17.602    0.0250000000000213
05:42:17.602    0.019999999999999574
05:42:17.602    0.020000000000003126
05:42:17.603    0.019999999999999574
05:42:17.603    0.019999999999999574
05:42:17.603    0.015000000000000568
05:42:17.604    0.054999999999999716
```



```
05:42:33.296    0.22499999999999787
05:42:33.297    0.045000000000001705
05:42:33.298    0.02499999999999858
05:42:33.298    0.030000000000001137
05:42:33.299    0.015000000000000568
05:42:33.299    0.015000000000000568
05:42:33.299    0.015000000000000568
05:42:33.300    0.02499999999999858
05:42:33.300    0.015000000000000568
05:42:33.300    0.05000000000000426
```

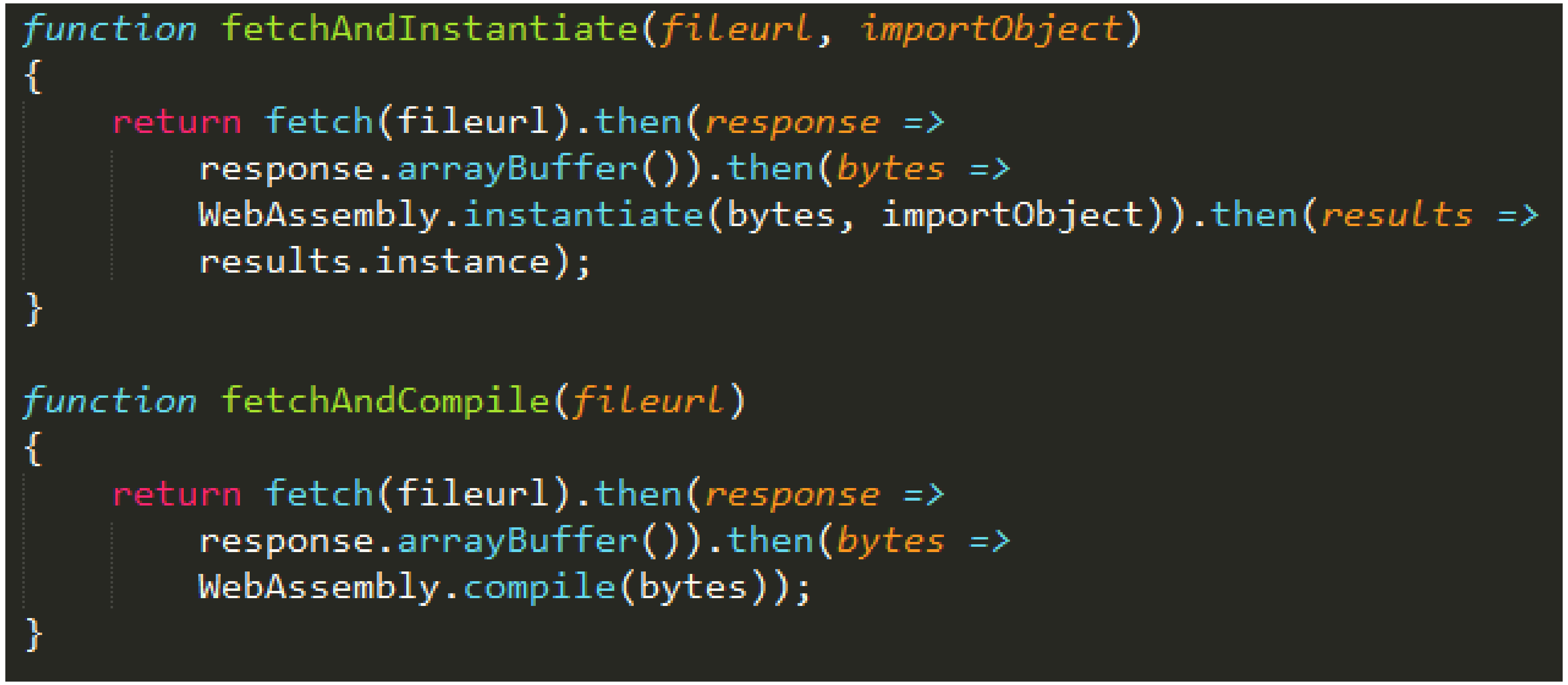


```
05:42:56.179    0.22500000000000142
05:42:56.181    0.09499999999999886
05:42:56.182    0.03999999999999915
05:42:56.183    0.054999999999999716
05:42:56.183    0.035000000000003695
05:42:56.184    0.03499999999999659
05:42:56.185    0.07499999999999574
05:42:56.186    0.07499999999999574
05:42:56.186    0.035000000000003695
05:42:56.188    0.0799999999999983
```



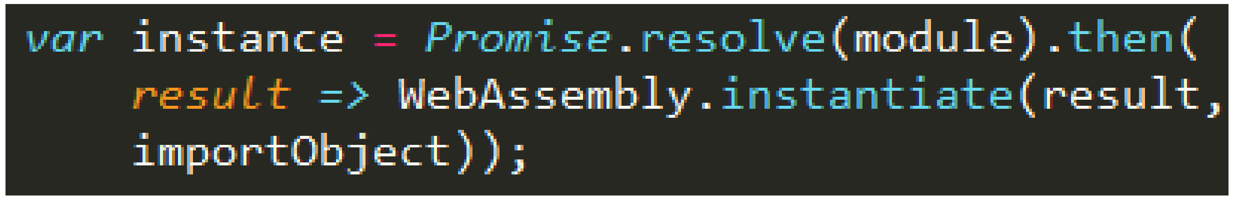
```
05:43:10.499    0.245000000000001
05:43:10.500    0.03999999999999915
05:43:10.501    0.019999999999999574
05:43:10.501    0.02499999999999858
05:43:10.501    0.015000000000000568
05:43:10.501    0.019999999999999574
05:43:10.502    0.015000000000000568
05:43:10.502    0.0250000000000213
05:43:10.502    0.014999999999997016
05:43:10.503    0.054999999999999716
```

Application of WASM

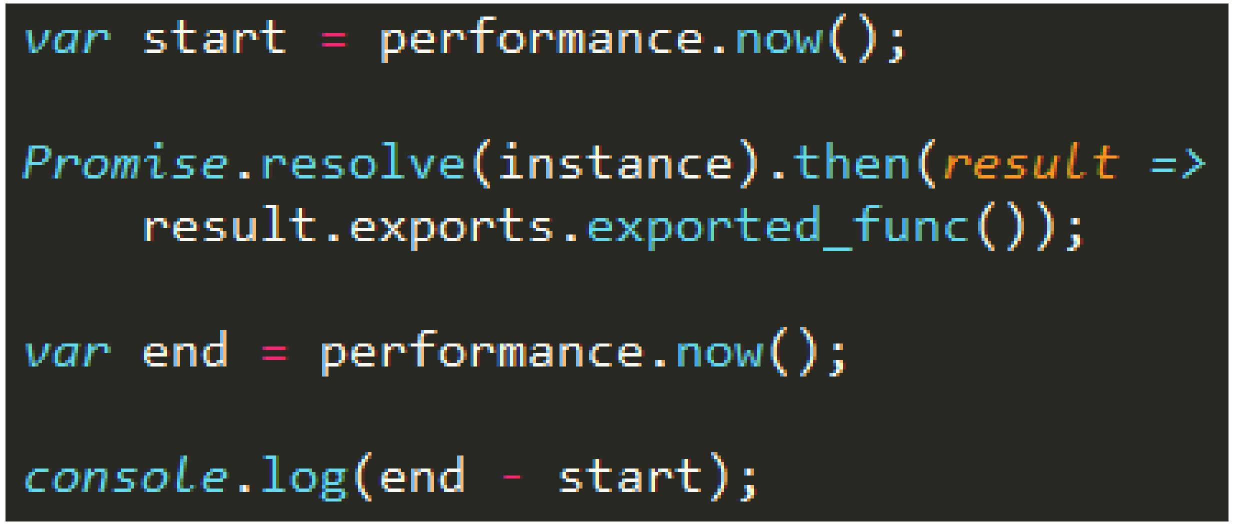


```javascript
function fetchAndInstantiate(fileurl, importObject)
{
    return fetch(fileurl).then(response =>
        response.arrayBuffer()).then(bytes =>
        WebAssembly.instantiate(bytes, importObject)).then(results =>
        results.instance);
}

function fetchAndCompile(fileurl)
{
    return fetch(fileurl).then(response =>
        response.arrayBuffer()).then(bytes =>
        WebAssembly.compile(bytes));
}
```

Application of WASM

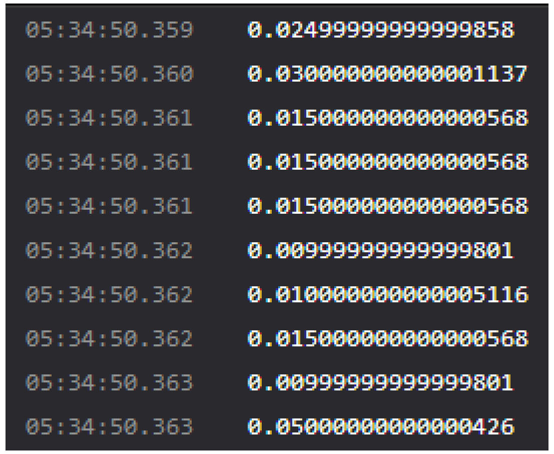


```
var instance = Promise.resolve(module).then(
    result => WebAssembly.instantiate(result,
    importObject));
```

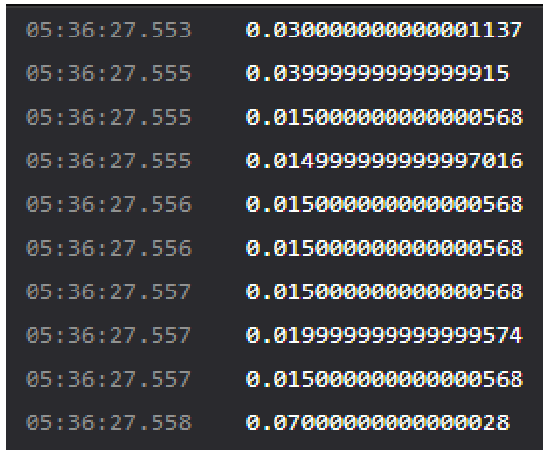


```
var start = performance.now();

Promise.resolve(instance).then(result =>
    result.exports.exported_func());

var end = performance.now();

console.log(end - start);
```

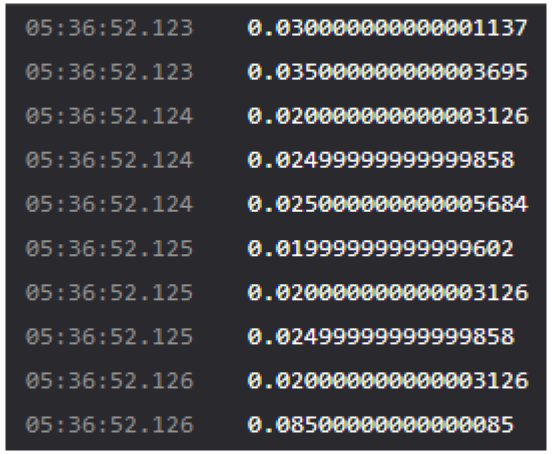
Application of WASM



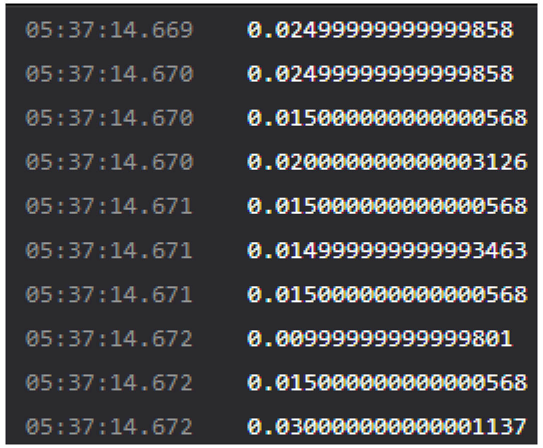
```
05:34:50.359    0.024999999999999858
05:34:50.360    0.030000000000001137
05:34:50.361    0.015000000000000568
05:34:50.361    0.015000000000000568
05:34:50.361    0.015000000000000568
05:34:50.362    0.009999999999999801
05:34:50.362    0.010000000000005116
05:34:50.362    0.015000000000000568
05:34:50.363    0.009999999999999801
05:34:50.363    0.05000000000000426
```



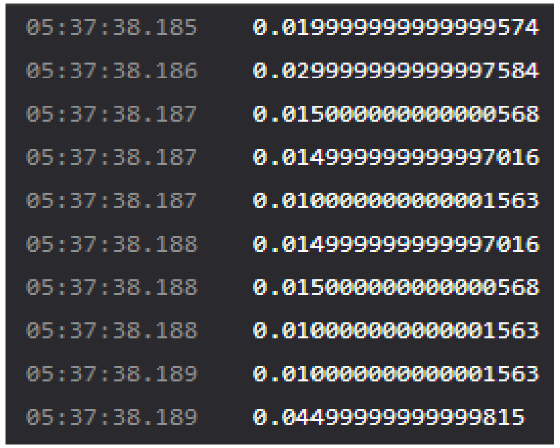
```
05:36:27.553    0.030000000000001137
05:36:27.555    0.039999999999999915
05:36:27.555    0.015000000000000568
05:36:27.555    0.014999999999997016
05:36:27.556    0.015000000000000568
05:36:27.556    0.015000000000000568
05:36:27.557    0.015000000000000568
05:36:27.557    0.019999999999999574
05:36:27.557    0.015000000000000568
05:36:27.558    0.07000000000000028
```



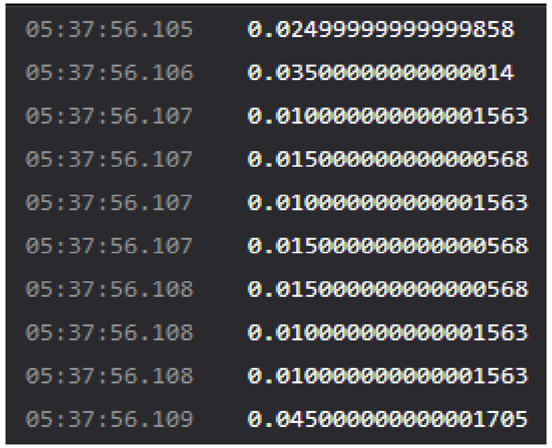
```
05:36:52.123    0.030000000000001137
05:36:52.123    0.035000000000003695
05:36:52.124    0.020000000000003126
05:36:52.124    0.024999999999999858
05:36:52.124    0.025000000000005684
05:36:52.125    0.019999999999999602
05:36:52.125    0.020000000000003126
05:36:52.125    0.024999999999999858
05:36:52.126    0.020000000000003126
05:36:52.126    0.08500000000000085
```



```
05:37:14.669    0.024999999999999858
05:37:14.670    0.024999999999999858
05:37:14.670    0.015000000000000568
05:37:14.670    0.020000000000003126
05:37:14.671    0.015000000000000568
05:37:14.671    0.014999999999993463
05:37:14.671    0.015000000000000568
05:37:14.672    0.009999999999999801
05:37:14.672    0.015000000000000568
05:37:14.672    0.030000000000001137
```

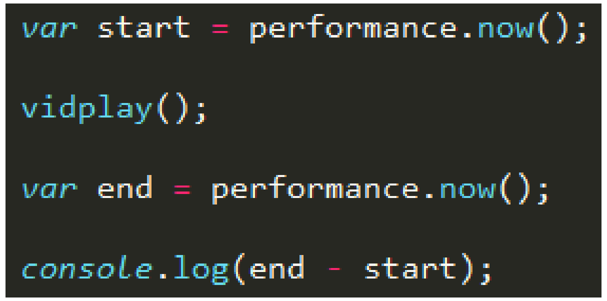


```
05:37:38.185    0.019999999999999574
05:37:38.186    0.029999999999997584
05:37:38.187    0.015000000000000568
05:37:38.187    0.014999999999997016
05:37:38.187    0.010000000000001563
05:37:38.188    0.014999999999997016
05:37:38.188    0.015000000000000568
05:37:38.188    0.010000000000001563
05:37:38.189    0.010000000000001563
05:37:38.189    0.044999999999999815
```

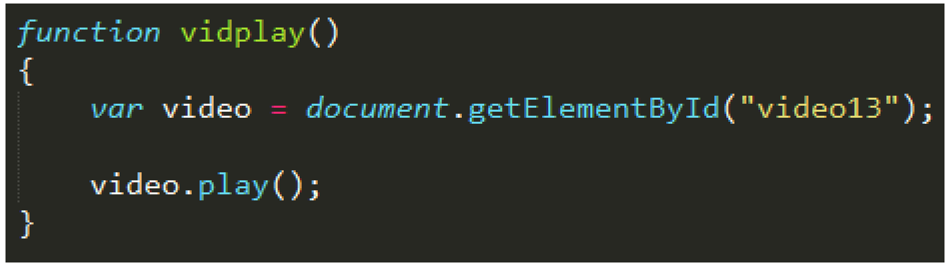


```
05:37:56.105    0.024999999999999858
05:37:56.106    0.03500000000000014
05:37:56.107    0.010000000000001563
05:37:56.107    0.015000000000000568
05:37:56.107    0.010000000000001563
05:37:56.107    0.015000000000000568
05:37:56.108    0.015000000000000568
05:37:56.108    0.010000000000001563
05:37:56.108    0.010000000000001563
05:37:56.109    0.045000000000001705
```

Application of WASM

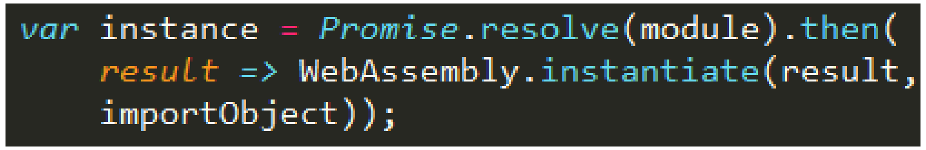


```
var start = performance.now();

vidplay();

var end = performance.now();

console.log(end - start);
```

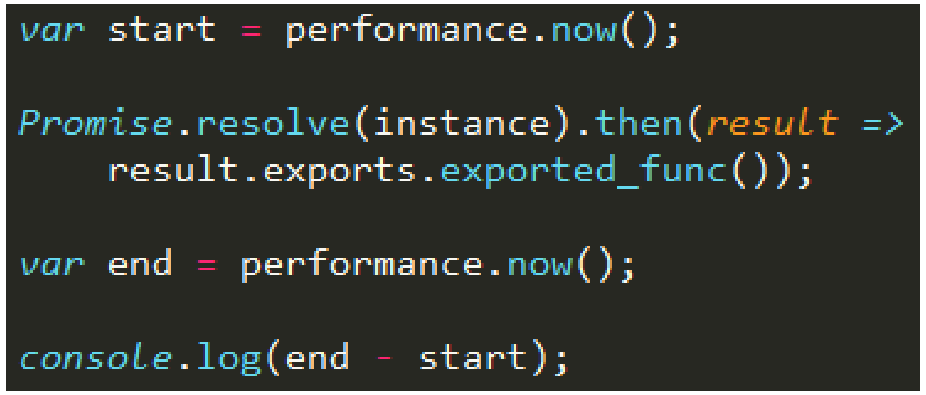


```
function vidplay()
{
    var video = document.getElementById("video13");

    video.play();
}
```

?



```
var instance = Promise.resolve(module).then(
    result => WebAssembly.instantiate(result,
    importObject));
```



```
var start = performance.now();

Promise.resolve(instance).then(result =>
    result.exports.exported_func());

var end = performance.now();

console.log(end - start);
```

Application of WASM - Conclusion

- Can manage suitable JS func with appropriate WASM code together
    - Expected to “stick” JS code with other languages supports WASM

- Shows equal or slightly better performance compared to pure JS
    - Can choose when to load / instantiate codes
    - Caching compiled modules via IndexedDB

- To complicated for general project
    - Promise, WAST(S-expression)
    - Necessity of compiler’s development / IDE

What have I learnt

- How to use Promise object
- Compilation pipeline of C/C++ to WASM
- How WASM works in web browser
- Complexity of raw s-expression

Todo

- Find a role in the project
- Concentrate on real implementation of web page / server app