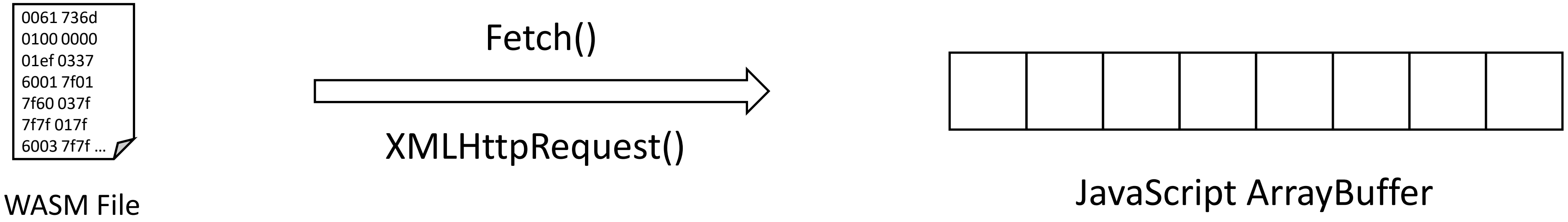
WebAssembly - II

Prerequisites / How to use it

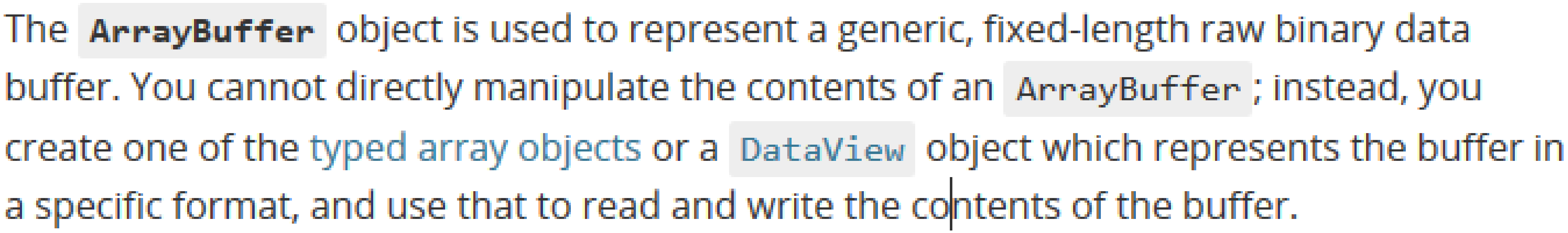
Contents

- How to load / compile / instantiate / run WASM code?
- Promise object
- Synchronous / Asynchronous?
- Future Works

How to load WASM code?

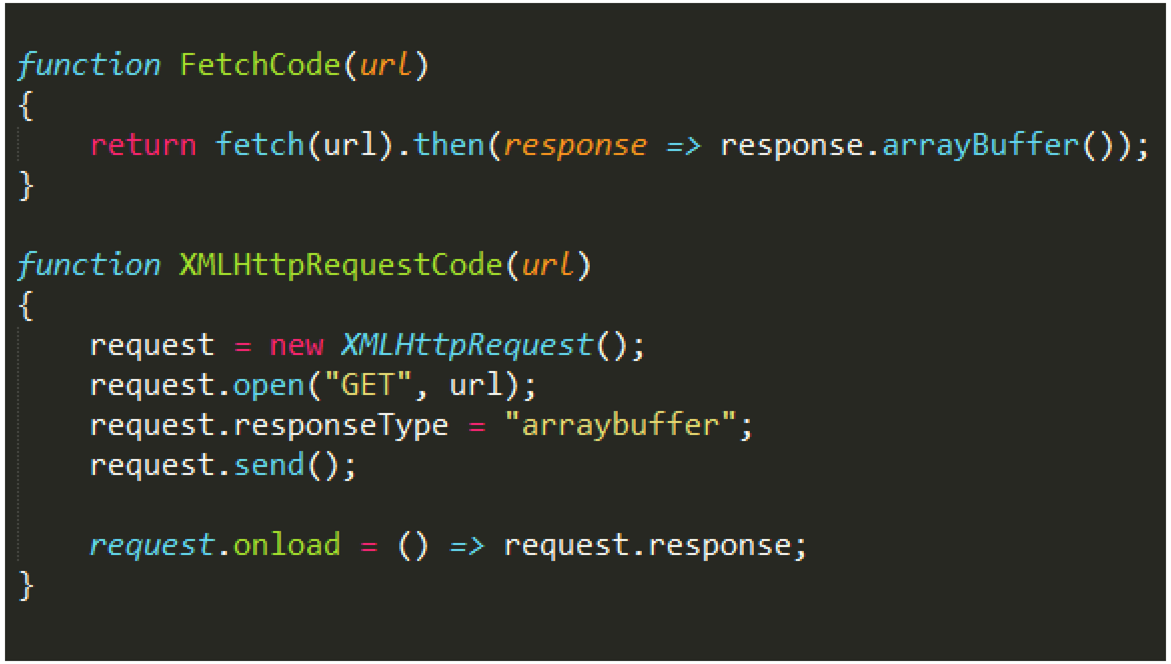


How to load WASM code?



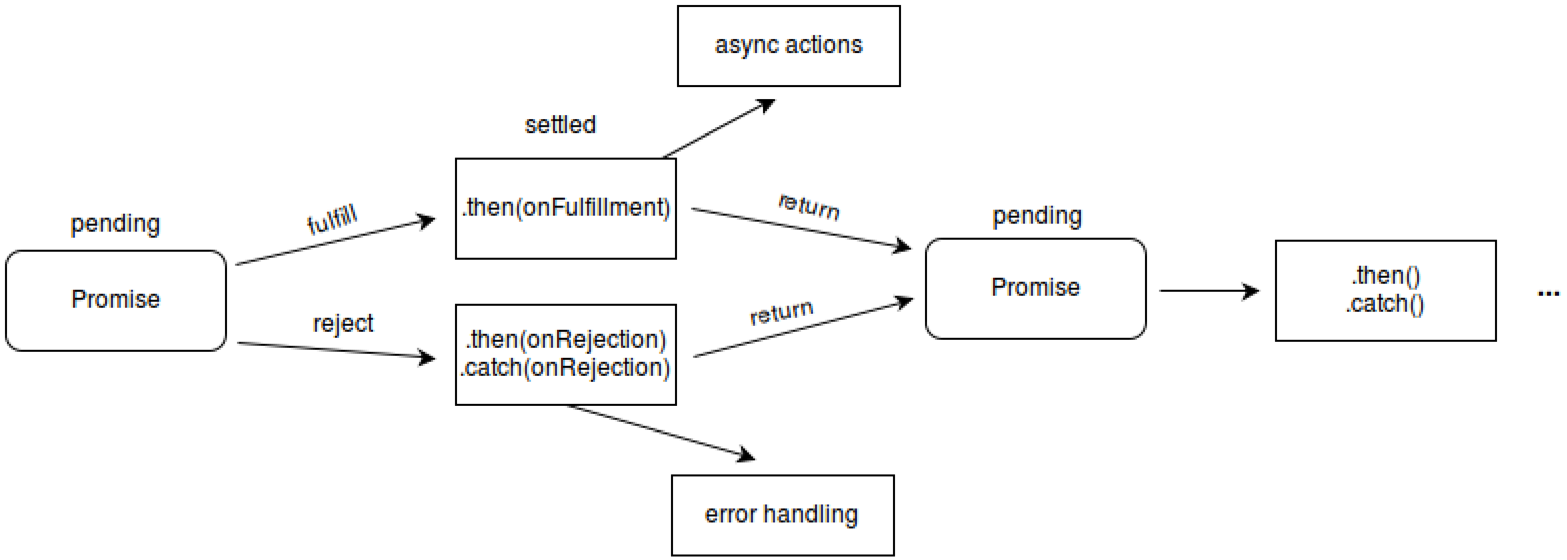
The **`ArrayBuffer`** object is used to represent a generic, fixed-length raw binary data buffer. You cannot directly manipulate the contents of an `ArrayBuffer`; instead, you create one of the typed array objects or a `DataView` object which represents the buffer in a specific format, and use that to read and write the contents of the buffer.

How to load WASM code?



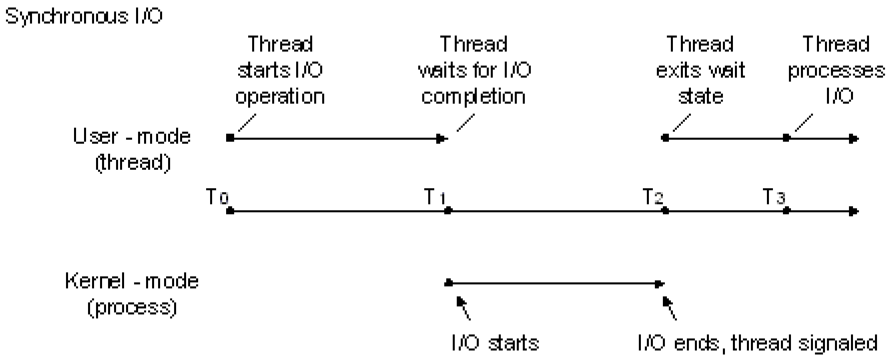
```javascript
function FetchCode(url)
{
    return fetch(url).then(response => response.arrayBuffer());
}

function XMLHttpRequestCode(url)
{
    request = new XMLHttpRequest();
    request.open("GET", url);
    request.responseType = "arraybuffer";
    request.send();

    request.onload = () => request.response;
}
```

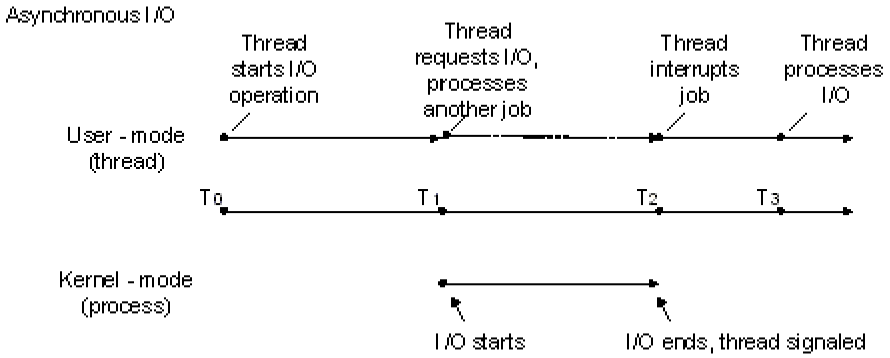
Promise object



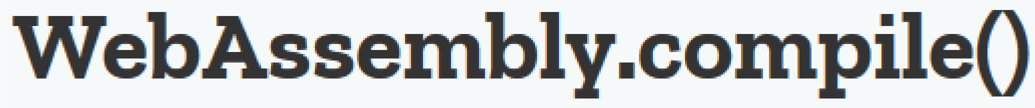
Essentially, a promise is a returned object to which you attach callbacks, instead of passing callbacks into a function.

Synchronous / Asynchronous

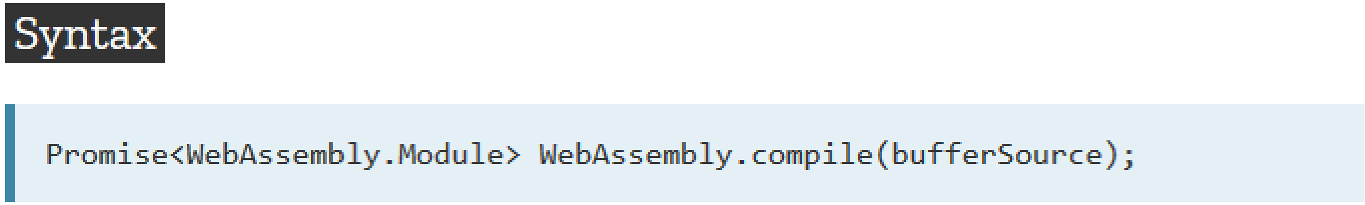

Synchronous I/O


Asynchronous I/O

How to compile WASM code?



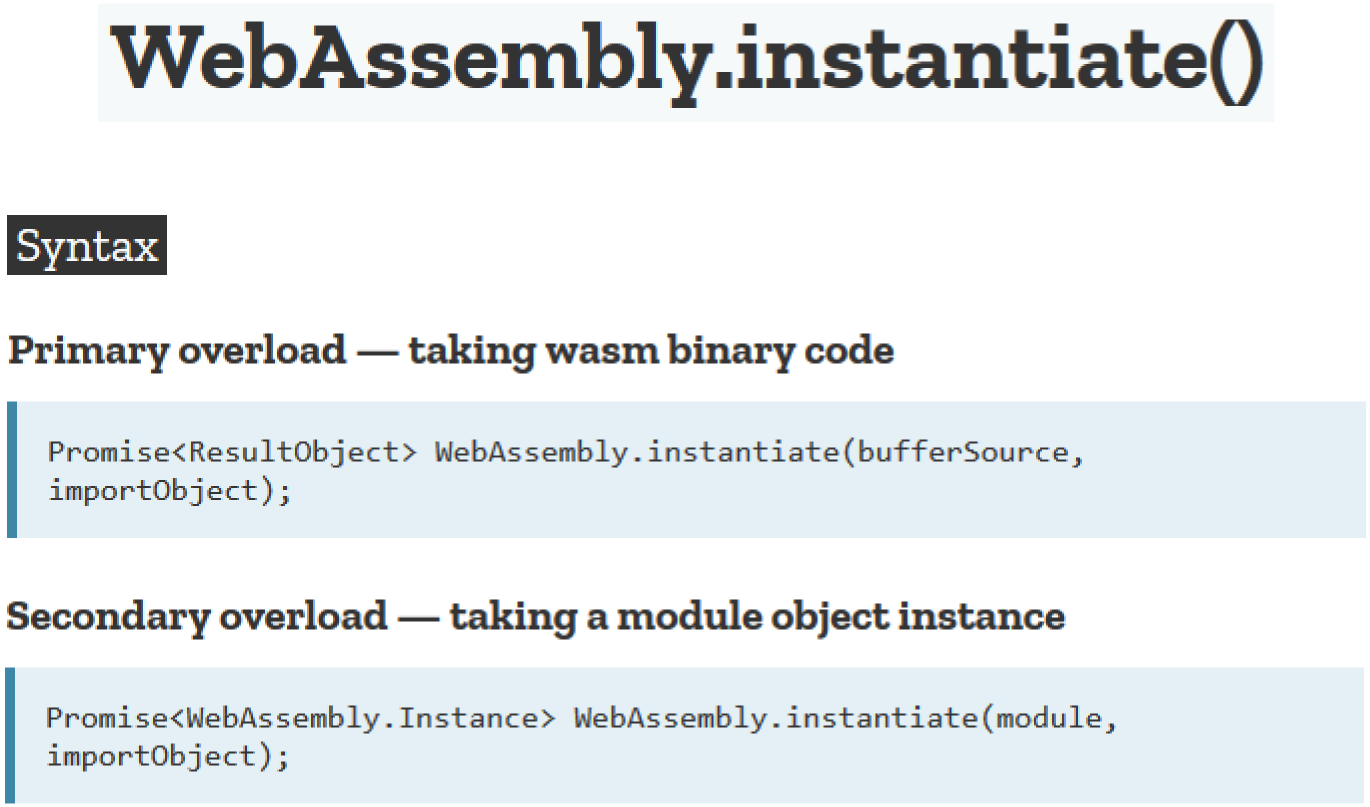
WebAssembly.compile()



Syntax

```
Promise<WebAssembly.Module> WebAssembly.compile(bufferSource);
```

How to instantiate WASM code?



WebAssembly.instantiate()

Syntax

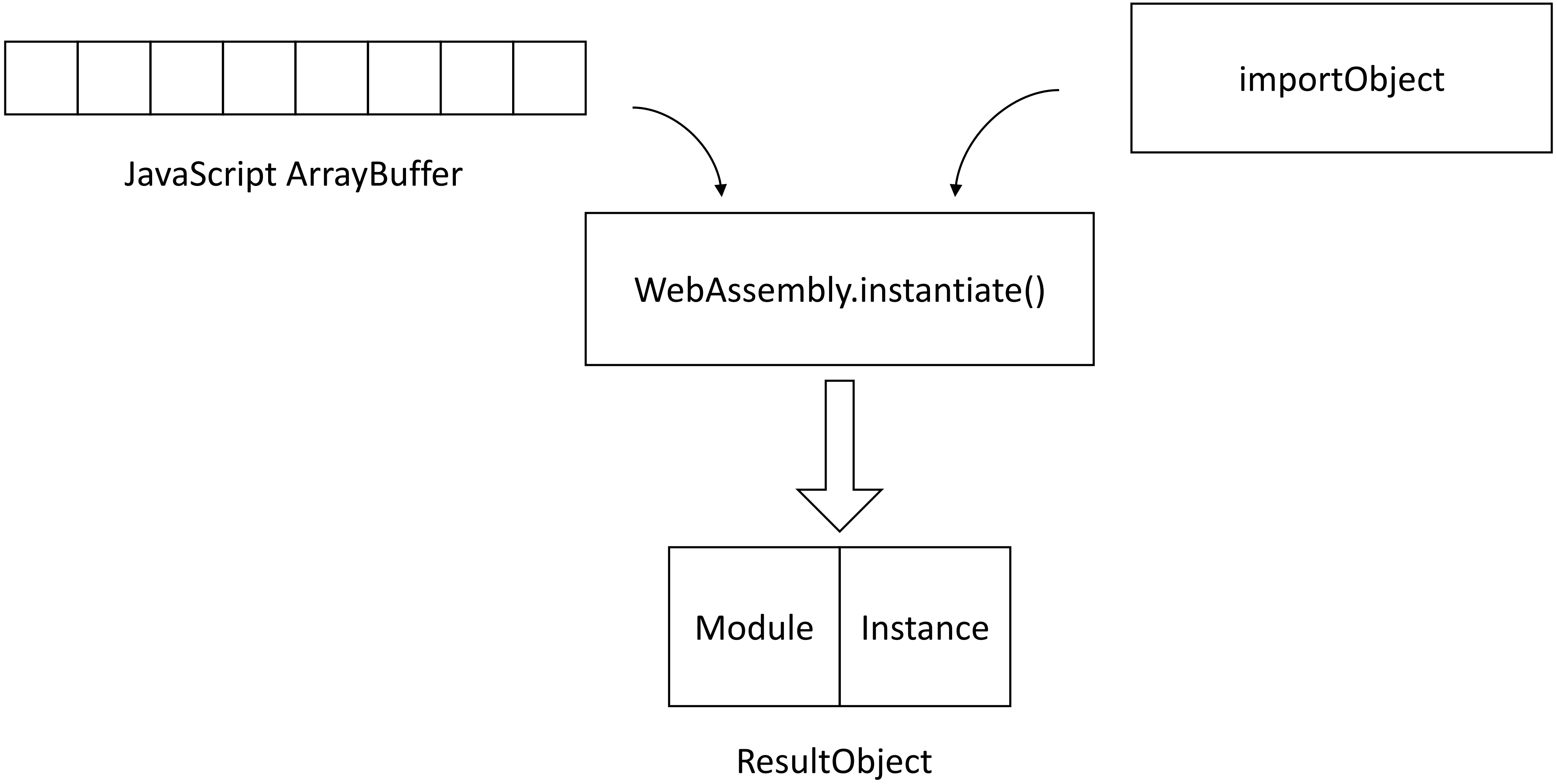
Primary overload — taking wasm binary code

```
Promise<ResultObject> WebAssembly.instantiate(bufferSource,
importObject);
```

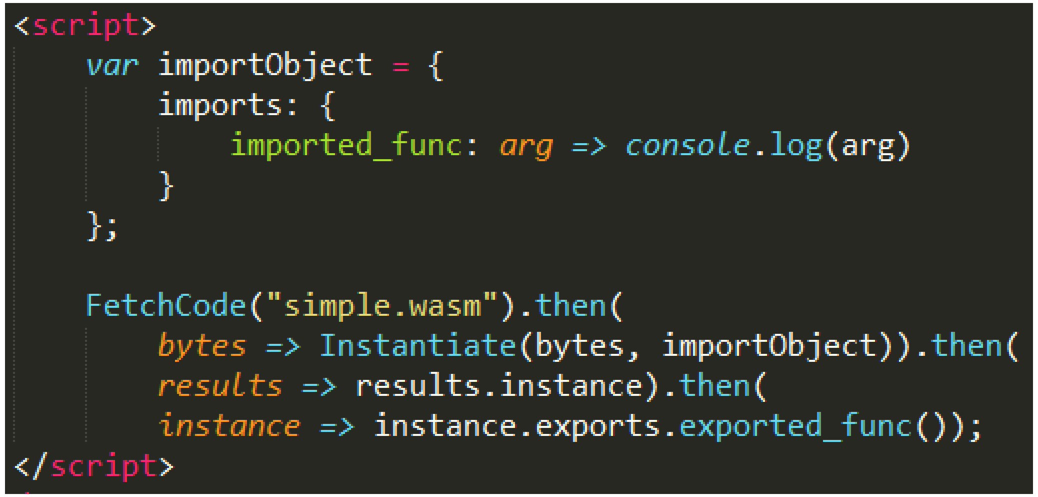
Secondary overload — taking a module object instance

```
Promise<WebAssembly.Instance> WebAssembly.instantiate(module,
importObject);
```

How to instantiate WASM code?



How to run WASM code?



```
<script>
    var importObject = {
        imports: {
            imported_func: arg => console.log(arg)
        }
    };

    FetchCode("simple.wasm").then(
        bytes => Instantiate(bytes, importObject)).then(
        results => results.instance).then(
        instance => instance.exports.exported_func());
</script>
```

Future Work

- Import / Export

- Import Object

- S-expression