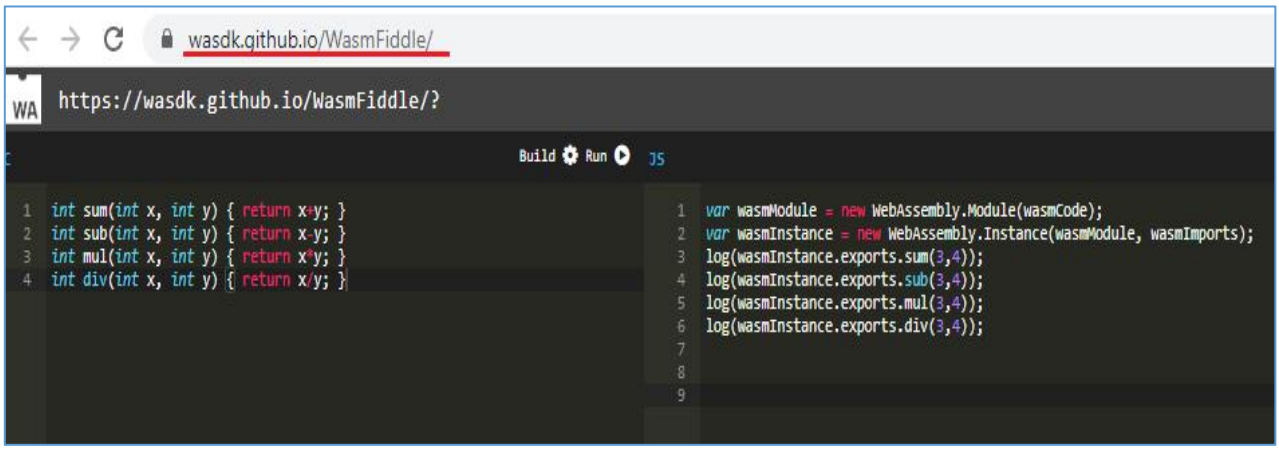
Лекция 29
ПСКП
ПОИТ-3

WebAssembly (WASM) в Node.js

1. **WASM: WebAssembly** – бинарный формат исполняемого файла, который может исполняться в JavaScript Engine (виртуальная стековая машина).
2. **WASM:** код быстрее, чем JS; поддерживается большинством браузеров; выполняется в sandbox; есть отладчики; открытый стандарт.

3. **WASM:**

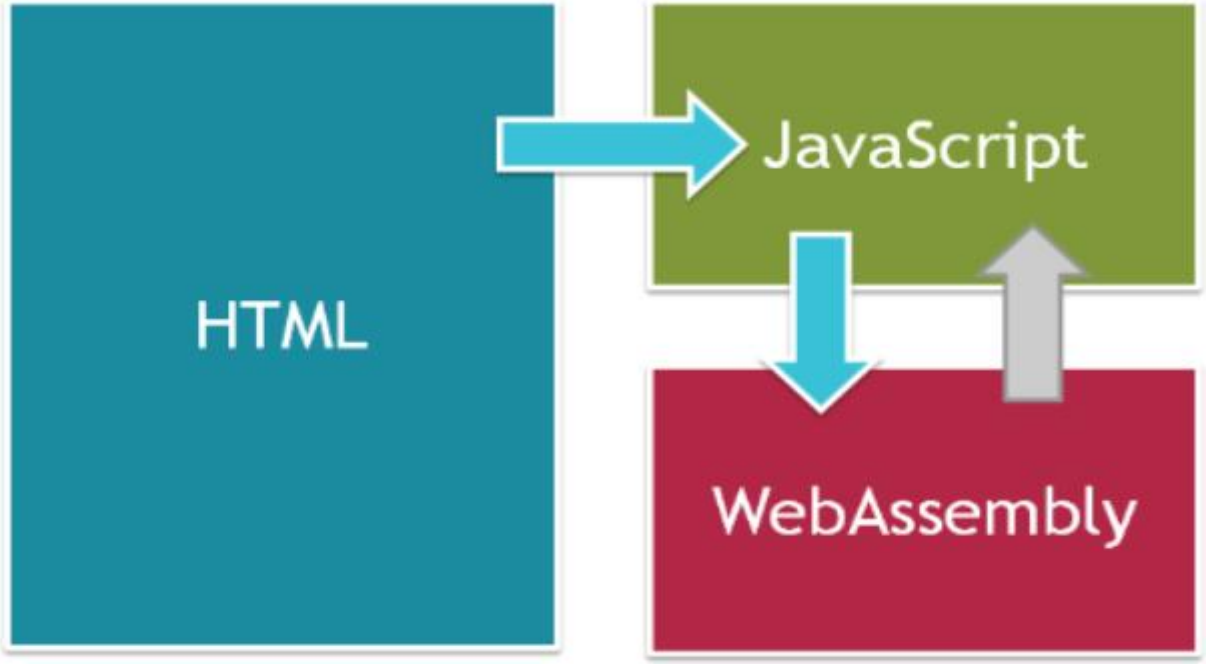
4. **WASM: WasmFiddle**

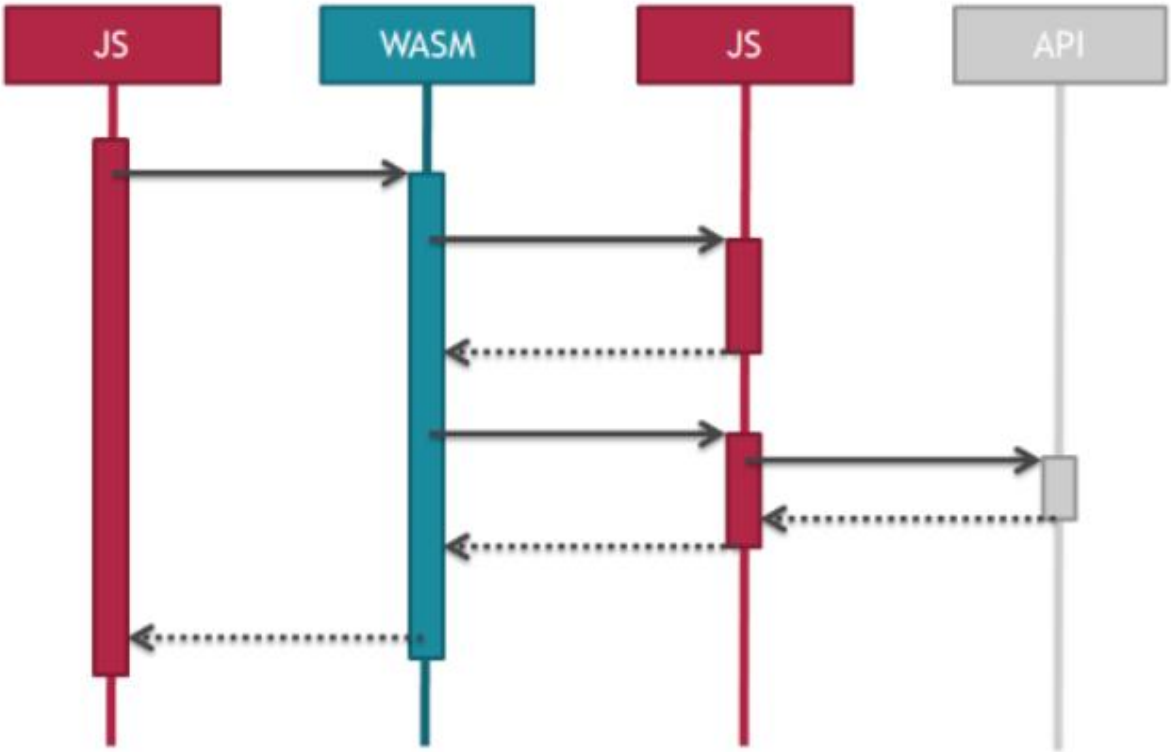


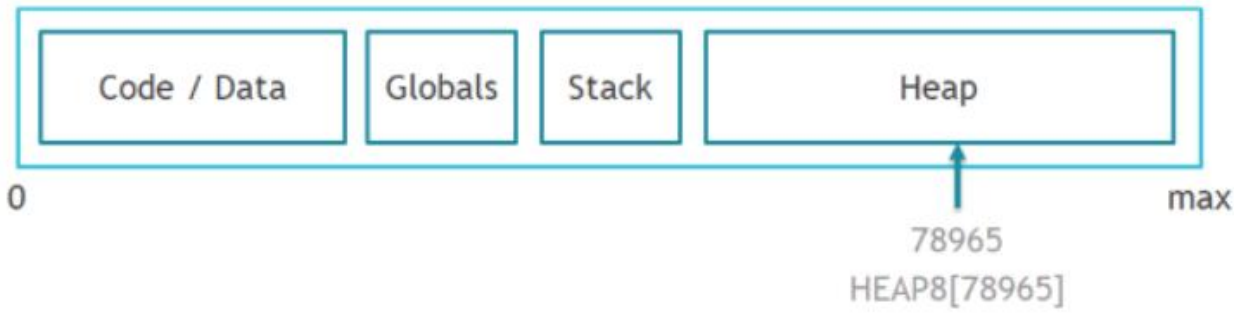
5. WASM: WASM/Browser

```html
<!DOCTYPE html>
<html>
    <head></head>
    <body>
        <h1>WASM</h1>
        <p> sum(3,4) = <span id="rsum"></span> </p>
        <p> sub(3,4) = <span id="rsub"></span> </p>
        <p> mul(3,4) = <span id="rmul"></span> </p>
        <p> div(3,4) = <span id="rdiv"></span> </p>

        <script>
            // int sum(int x, int y) { return x+y; }
            // int sub(int x, int y) { return x-y; }
            // int mul(int x, int y) { return x*y; }
            // int div(int x, int y) { return x/y; }

            let wasmCode = new Uint8Array([0,97,115,109,1,0,0,0,1,135,128,128,128,0,1,96,2,127,127,1,127,3,133,128,
                                           128,128,0,4,0,0,0,0,4,132,128,128,128,0,1,112,0,0,5,131,128,128,128,0,1,0,
                                           1,6,129,128,128,128,0,0,7,162,128,128,128,0,5,6,109,101,109,111,114,121,2,
                                           0,3,115,117,109,0,0,3,115,117,98,0,1,3,109,117,108,0,2,3,100,105,118,0,3,10,
                                           177,128,128,128,0,4,135,128,128,128,0,0,32,1,32,0,106,11,135,128,128,128,0,0,
                                           32,0,32,1,107,11,135,128,128,128,0,0,32,1,32,0,108,11,135,128,128,128,0,0,32,
                                           0,32,1,109,11]);

            let wasmImports  = {};
            let wasmModule   = new WebAssembly.Module(wasmCode);
            let wasmInstance = new WebAssembly.Instance(wasmModule, wasmImports);
            rsum.innerHTML = wasmInstance.exports.sum(3,4);
            rsub.innerHTML = wasmInstance.exports.sub(3,4);
            rmul.innerHTML = wasmInstance.exports.mul(3,4);
            rdiv.innerHTML = wasmInstance.exports.div(3,4);

        </script>
    </body>
</html>
```

6. **WASM: WASM/Browser**







7. **WASM:** emcc – компилятор c->wasm, wasm

установка emcc https://emscripten.org/docs/getting_started/downloads.html

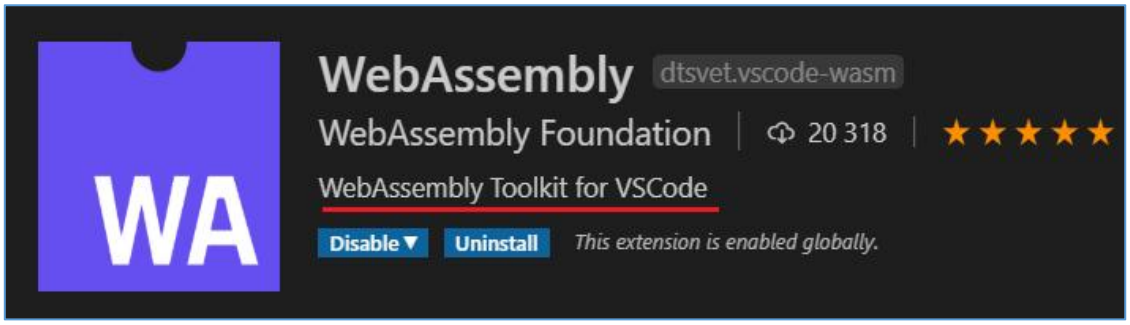
```
// p.c
#include <emscripten/emscripten.h>

#ifdef __cplusplus
    extern "C" {
#endif


int EMSCRIPTEN_KEEPALIVE sum(int x, int y){return x+y;}
int EMSCRIPTEN_KEEPALIVE sub(int x, int y){return x-y;}
int EMSCRIPTEN_KEEPALIVE mul(int x, int y){return x*y;}

#ifdef __cplusplus
}
#endif
```

Администратор: C:\Windows\System32\cmd.exe

```
D:\PSCA\Lec32_wasm>emcc  -O3 -o p.wasm -s WASM=1 p.c

D:\PSCA\Lec32_wasm>
```

(D:) › PSCA › Lec32_wasm ›

Имя	Дата изменения	Тип	Размер
emsdk	05.05.2020 21:36	Папка с файлами	
public	06.05.2020 23:17	Папка с файлами	
32-01.js	06.05.2020 1:14	JetBrains WebStorm	1 КБ
emcchelp.txt	06.05.2020 23:09	Текстовый докум...	24 КБ
env.bat	05.05.2020 22:33	Пакетный файл ...	2 КБ
index.html	06.05.2020 1:14	Chrome HTML Do...	2 КБ
p.c	07.05.2020 1:08	C Source	1 КБ
p.js	07.05.2020 0:03	JetBrains WebStorm	12 КБ
p.wasm	07.05.2020 1:08	Файл "WASM"	1 КБ
prompt.txt	07.05.2020 0:25	Текстовый докум...	1 КБ

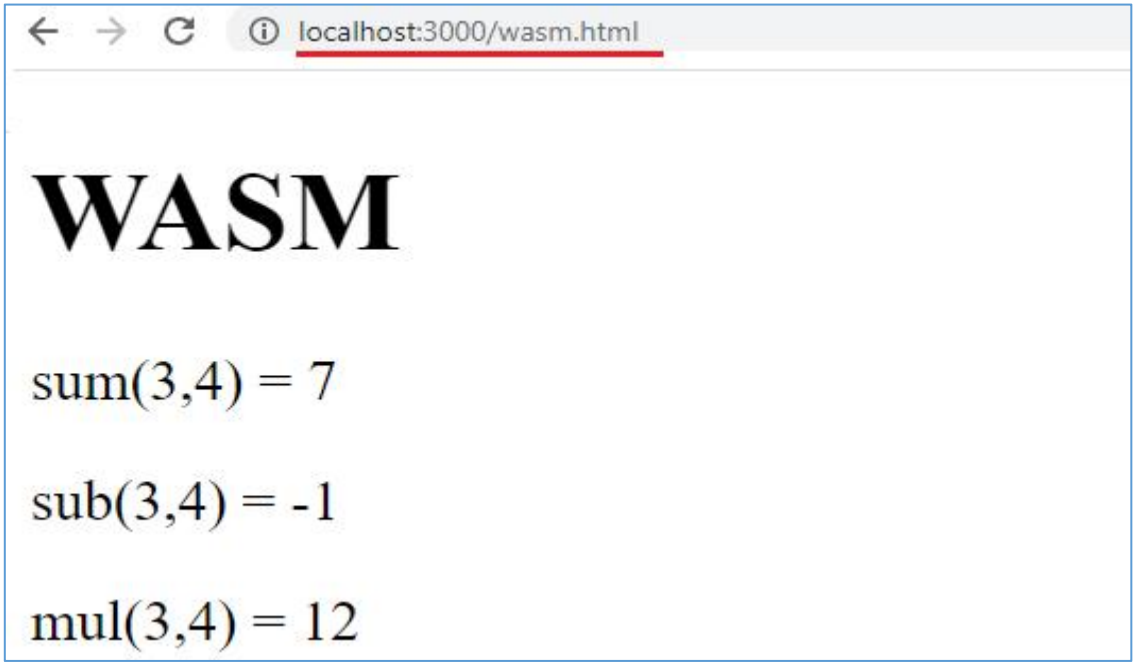



```
(module
  (type $t0 (func (param i32 i32) (result i32)))
  (type $t1 (func))
  (func $_start (type $t1)
    nop)
  (func $mul (type $t0) (param $p0 i32) (param $p1 i32) (result i32)
    local.get $p0
    local.get $p1
    i32.mul)
  (func $sub (type $t0) (param $p0 i32) (param $p1 i32) (result i32)
    local.get $p0
    local.get $p1
    i32.sub)
  (func $sum (type $t0) (param $p0 i32) (param $p1 i32) (result i32)
    local.get $p0
    local.get $p1
    i32.add)
  (memory $memory 256 256)
  (export "memory" (memory 0))
  (export "sum" (func $sum))
  (export "sub" (func $sub))
  (export "mul" (func $mul))
  (export "_start" (func $_start))
  (data $d0 (i32.const 1536) "\a0\06P"))
```

8. **WASM:** node/browser, wasm, js, fetch

```
const express = require("express");  // npm install express
const app = express();

app.use('/',express.static('public'));

app.use((req, res, next)=>{console.log('handler 02'); next();});

app.listen(3000,()=>console.log('Start server, port:', 3000));
```

```html
<!DOCTYPE html>
<html>
    <head></head>
    <body>
        <h1>WASM</h1>
        <p> sum(3,4) = <span id="rsum"></span> </p>
        <p> sub(3,4) = <span id="rsub"></span> </p>
        <p> mul(3,4) = <span id="rmul"></span> </p>

        <script>

            (async () => {
                const res =  fetch('http://localhost:3000/p.wasm');
                const {module, instance}  = await WebAssembly.instantiateStreaming(res);
                rsum.innerHTML = instance.exports.sum(3,4);
                rsub.innerHTML = instance.exports.sub(3,4);
                rmul.innerHTML = instance.exports.mul(3,4);
            })();

        </script>

    </body>
</html>
```



9. **WASM:** browser, wasm, js, Unit8Array

p.wasm.hexdump

```
Offset: 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
00000000: 00 61 73 6D 01 00 00 00 01 0A 02 60 02 7F 7F 01    .asm.......`....
00000010: 7F 60 00 00 03 05 04 01 00 00 00 05 06 01 01 80    .`..............
00000020: 02 80 02 07 25 05 06 6D 65 6D 6F 72 79 02 00 03    ....%..memory...
00000030: 73 75 6D 00 03 03 73 75 62 00 02 03 6D 75 6C 00    sum...sub...mul.
00000040: 01 06 5F 73 74 61 72 74 00 00 0A 1D 04 03 00 01    .._start........
00000050: 0B 07 00 20 00 20 01 6C 0B 07 00 20 00 20 01 6B    .......l.......k
00000060: 0B 07 00 20 00 20 01 6A 0B 0B 0A 01 00 41 80 0C    .......j.....A..
00000070: 0B 03 A0 06 50                                     ....P
```

```html
<!DOCTYPE html>
<html>
    <head></head>
    <body>
        <h1>WASM</h1>
        <p> sum(3,4) = <span id="rsum"></span> </p>
        <p> sub(3,4) = <span id="rsub"></span> </p>
        <p> mul(3,4) = <span id="rmul"></span> </p>
        <script>

            let wasmCode = new Uint8Array(
            [0x00,0x61,0x73,0x6D,0x01,0x00,0x00,0x00,0x01,0x0A,0x02,0x60,0x02,0x7F,0x7F,0x01,
             0x7F,0x60,0x00,0x00,0x03,0x05,0x04,0x01,0x00,0x00,0x00,0x05,0x06,0x01,0x01,0x80,
             0x02,0x80,0x02,0x07,0x25,0x05,0x06,0x6D,0x65,0x6D,0x6F,0x72,0x79,0x02,0x00,0x03,
             0x73,0x75,0x6D,0x00,0x03,0x03,0x73,0x75,0x62,0x00,0x02,0x03,0x6D,0x75,0x6C,0x00,
             0x01,0x06,0x5F,0x73,0x74,0x61,0x72,0x74,0x00,0x00,0x0A,0x1D,0x04,0x03,0x00,0x01,
             0x0B,0x07,0x00,0x20,0x00,0x20,0x01,0x6C,0x0B,0x07,0x00,0x20,0x00,0x20,0x01,0x6B,
             0x0B,0x07,0x00,0x20,0x00,0x20,0x01,0x6A,0x0B,0x0B,0x0A,0x01,0x00,0x41,0x80,0x0C,
             0x0B,0x03,0xA0,0x06,0x50]);

            let wasmImports  = {};
            let wasmModule   = new WebAssembly.Module(wasmCode);
            let wasmInstance = new WebAssembly.Instance(wasmModule, wasmImports);
            rsum.innerHTML = wasmInstance.exports.sum(3,4);
            rsub.innerHTML = wasmInstance.exports.sub(3,4);
            rmul.innerHTML = wasmInstance.exports.mul(3,4);


        </script>
    </body>
</html>
```

localhost:3000/wasm1.html

WASM

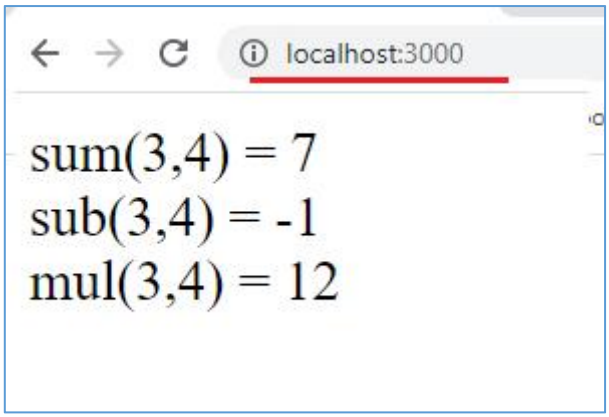
sum(3,4) = 7

sub(3,4) = -1

mul(3,4) = 12

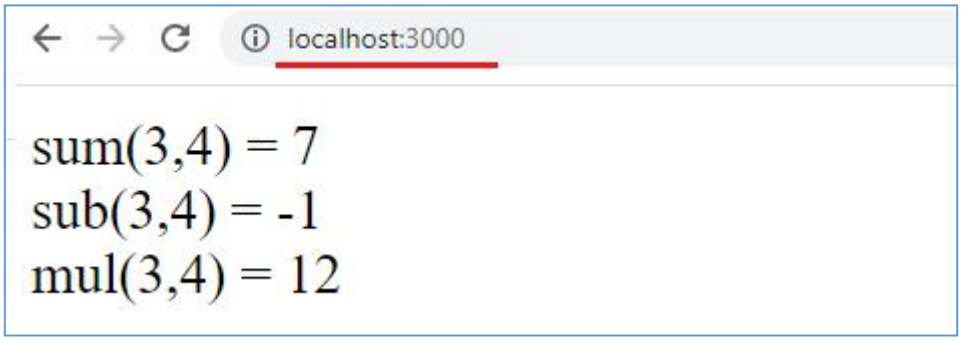
10. **WASM:** node.js, wasm, Unit8Array

```
const express = require("express");
const app =  express();

let wasmCode = new Uint8Array(
    [0x00,0x61,0x73,0x6D,0x01,0x00,0x00,0x00,0x01,0x0A,0x02,0x60,0x02,0x7F,0x7F,0x01,
     0x7F,0x60,0x00,0x00,0x03,0x05,0x04,0x01,0x00,0x00,0x00,0x05,0x06,0x01,0x01,0x80,
     0x02,0x80,0x02,0x07,0x25,0x05,0x06,0x6D,0x65,0x6D,0x6F,0x72,0x79,0x02,0x00,0x03,
     0x73,0x75,0x6D,0x00,0x03,0x03,0x73,0x75,0x62,0x00,0x02,0x03,0x6D,0x75,0x6C,0x00,
     0x01,0x06,0x5F,0x73,0x74,0x61,0x72,0x74,0x00,0x00,0x0A,0x1D,0x04,0x03,0x00,0x01,
     0x0B,0x07,0x00,0x20,0x00,0x20,0x01,0x6C,0x0B,0x07,0x00,0x20,0x00,0x20,0x01,0x6B,
     0x0B,0x07,0x00,0x20,0x00,0x20,0x01,0x6A,0x0B,0x0B,0x0A,0x01,0x00,0x41,0x80,0x0C,
     0x0B,0x03,0xA0,0x06,0x50]);
let wasmImports  = {};
let wasmModule   = new WebAssembly.Module(wasmCode);
let wasmInstance = new WebAssembly.Instance(wasmModule, wasmImports);


app.get('/', (req, res, next)=>{

    res.type('html').send(
                `sum(3,4) = ${wasmInstance.exports.sum(3,4)} <br/>`+
                `sub(3,4) = ${wasmInstance.exports.sub(3,4)} <br/>`+
                `mul(3,4) = ${wasmInstance.exports.mul(3,4)}`
                );
})


app.listen(3000,()=>console.log('Start server, port:', 3000));
```



11. **WASM:** node.js, wasm, fs

```
const express = require('express');
const fs      = require('fs');
const app     = express();

let wasmCode =    fs.readFileSync('public/p.wasm');
console.log(wasmCode);
let wasmImports  = {};
let wasmModule   = new WebAssembly.Module(wasmCode);
let wasmInstance = new WebAssembly.Instance(wasmModule, wasmImports);

app.get('/', (req, res, next)=>{
    res.type('html').send(
                        `sum(3,4) = ${wasmInstance.exports.sum(3,4)} <br/>`+
                        `sub(3,4) = ${wasmInstance.exports.sub(3,4)} <br/>`+
                        `mul(3,4) = ${wasmInstance.exports.mul(3,4)}`
                       );
})

app.listen(3000,()=>console.log('Start server, port:', 3000));
```



12. **WASM:**
13. **WASM:**
14. **WASM:**
15.
16.
17.
18.
19. **WASM:**

https://habr.com/ru/post/446764/
https://habr.com/ru/post/342180/
https://wasdk.github.io/WasmFiddle/

https://developers.google.com/web/updates/2018/04/loading-wasm
https://emscripten.org/docs/index.html

20. **WASM:**
21. **WASM:**
22. **WASM:**
23. **WASM:**
24. **WASM:**
25. **WASM:**
26. **WASM:**
27. **WASM:**
28. **WASM:**
29. **WASM:**
30. **WASM:**
31. **WASM:**
32. **WASM:**
33. **WASM:**
34. **WASM:**
35. **WASM:**
36. **WASM:**

37.
38.
39.