Яндекс Карты

WebAssembly без купюр

Андрей Роенко, руководитель группы разработки JavaScript API Яндекс.Карт

План

- Опыт Яндекс.Карт
- WebAssembly
- Как устроен WebAssembly?
- Как этого хватает?
- Программируем на WebAssembly
- Будущее
- FAQ
- Не только web

Опыт Яндекс.Карт

Поиск мест и адресов

Москва 9°C

Еда

Продукты

Кино

Гостиницы

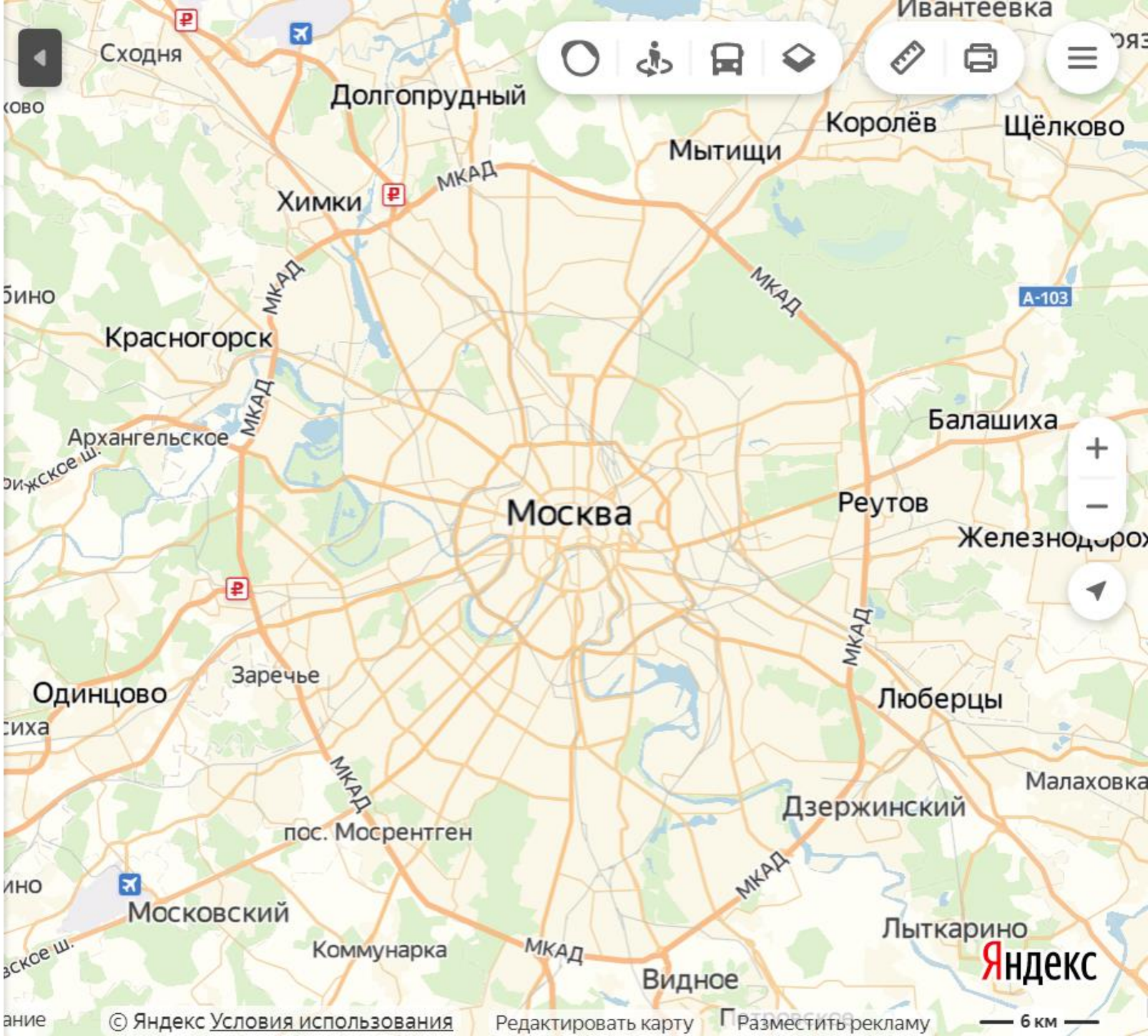
АЗС

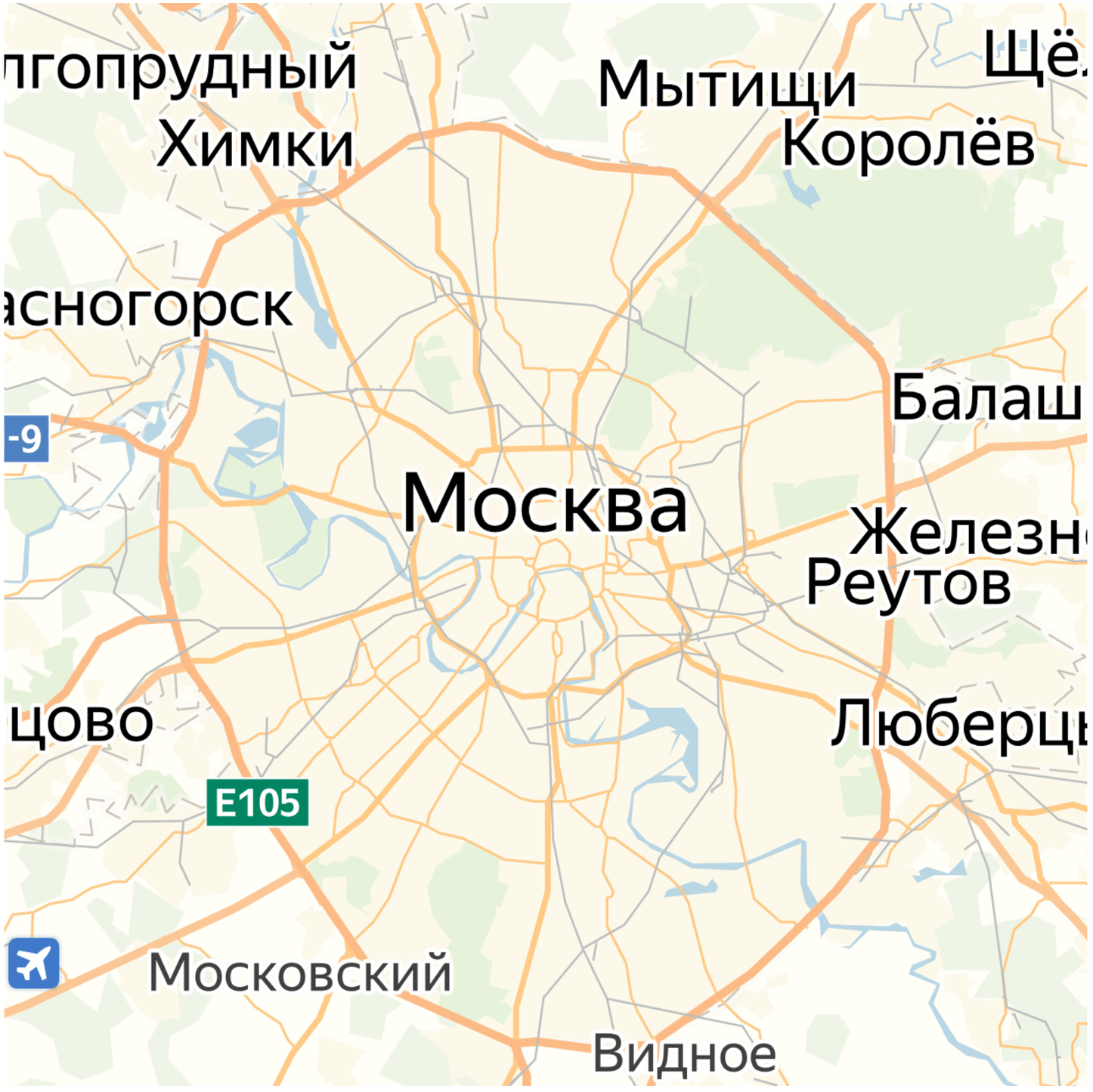
Аптеки

Бургер Кинг

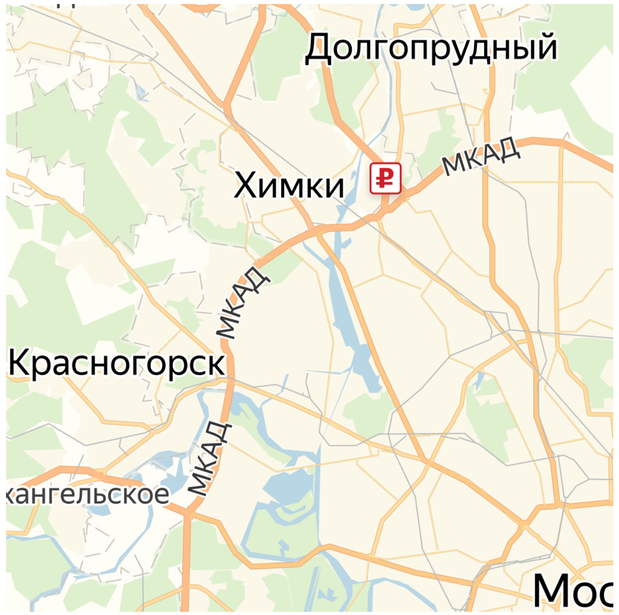
Mastercard

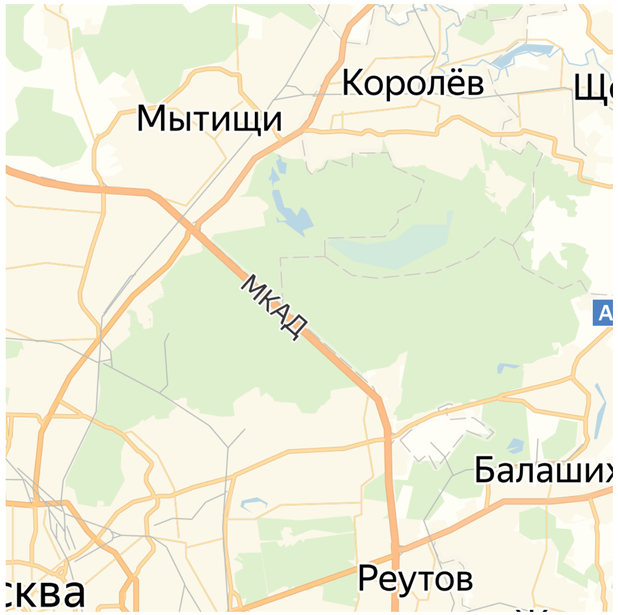
Посмотрите, как будут выглядеть новые районы Москвы и МО

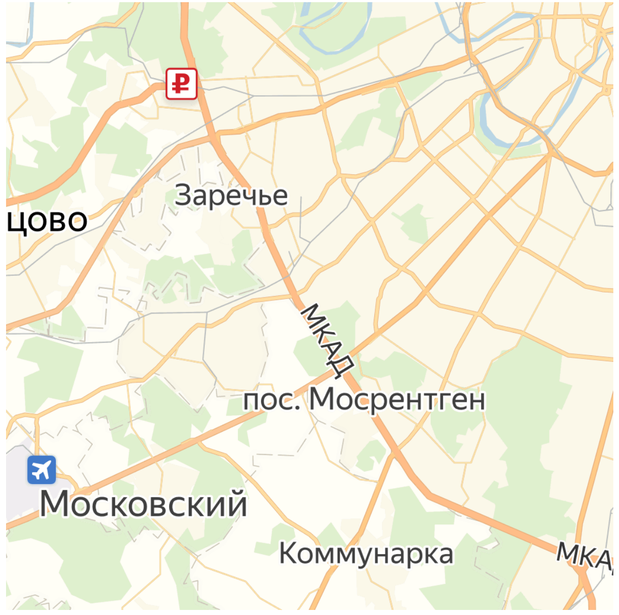


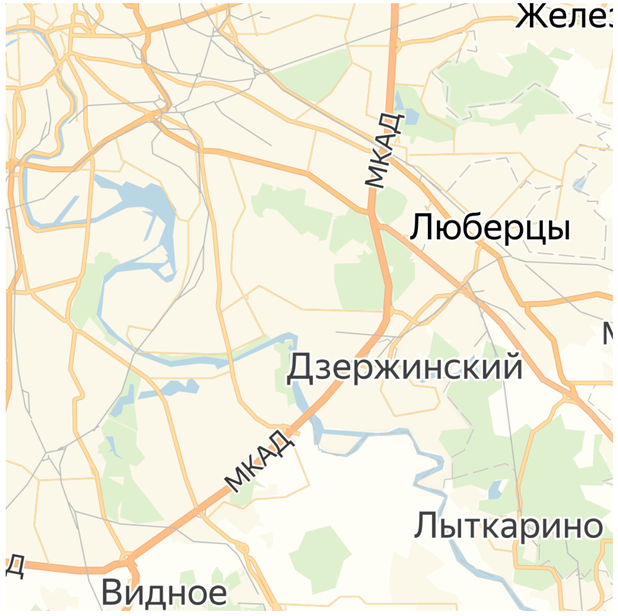

Мытищи
Щё
Химки
Королёв
Балаш
-9
Москва
Железн
Реутов
Люберц
E105
Московский
Видное

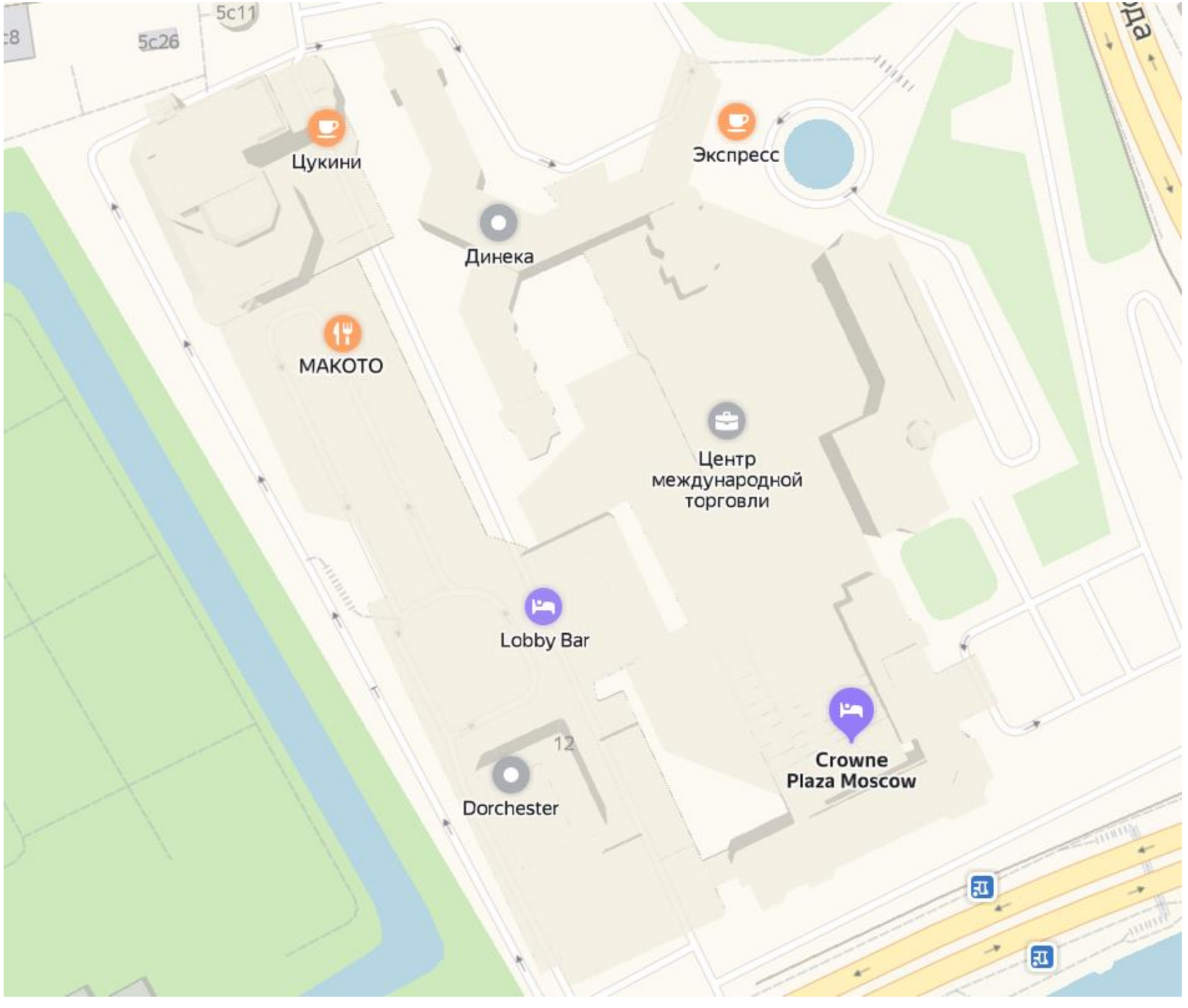

лгопрудный
Химки
Мытищи
Щё
Королёв
асногорск
-9
Балаш
Москва
Железн
Реутов
цово
Люберц
E105
Московский
Видное

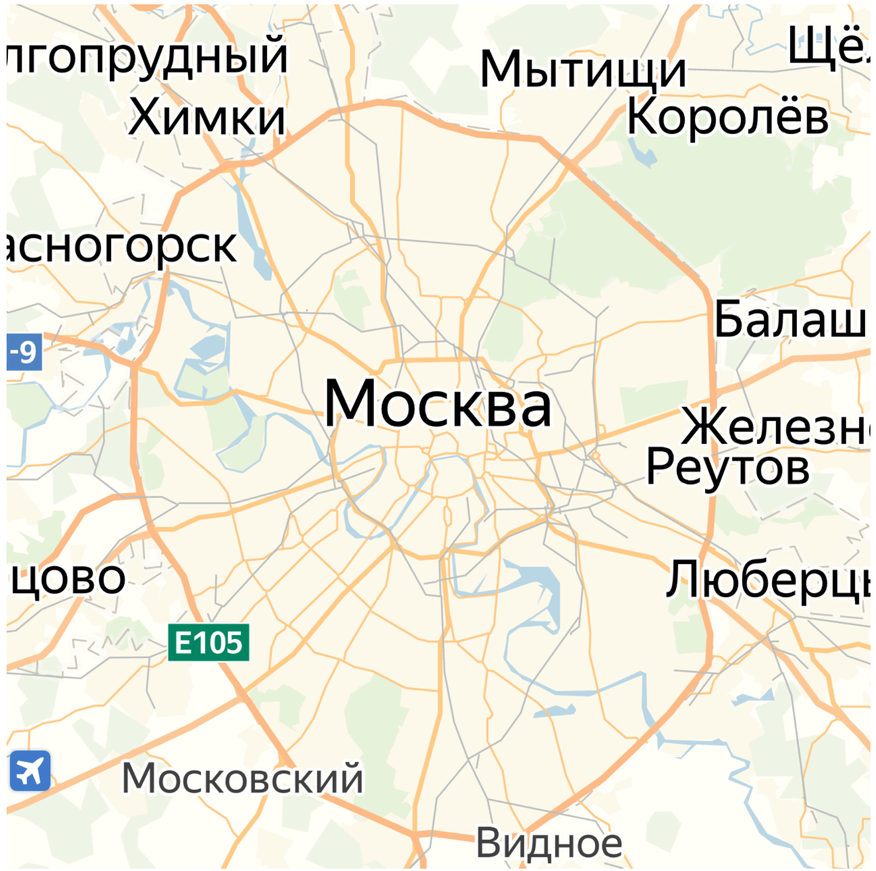

Долгопрудный
Химки
МКАД
МКАД
Красногорск
МКАД
кангельское
Мос


Королёв
Мытищи
МКАД
Балаших
Реутов
сква


Заречье
цово
МКАД
пос. Мосрентген
Московский
Коммунарка
МКА


Желе
МКАД
Люберцы
Дзержинский
МКАД
Лыткарино
Видное


5с11
5с26
Цукини
Экспресс
Динека
МАКОТО
Центр
международной
торговли
Lobby Bar
12
Dorchester
Crowne
Plaza Moscow

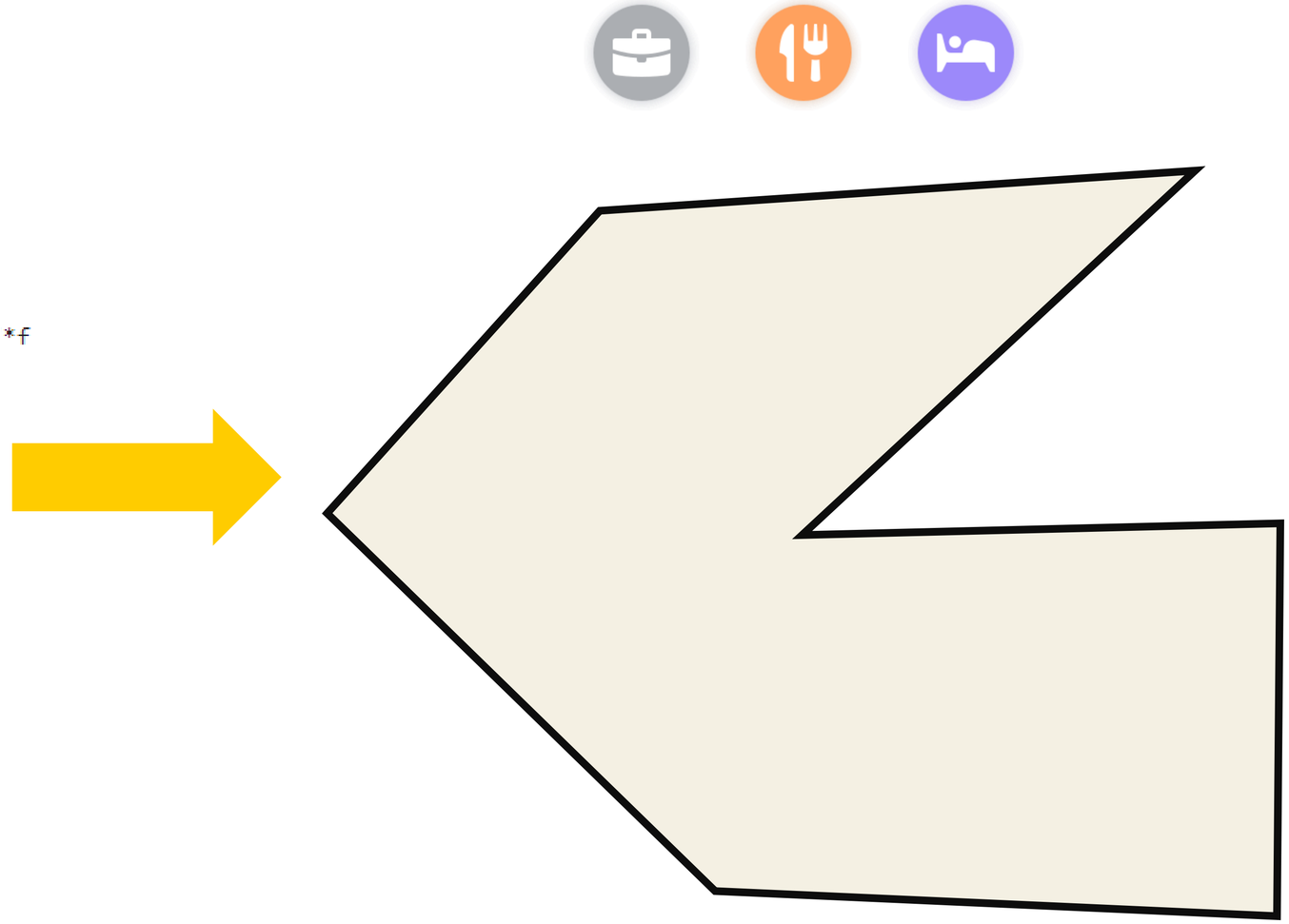


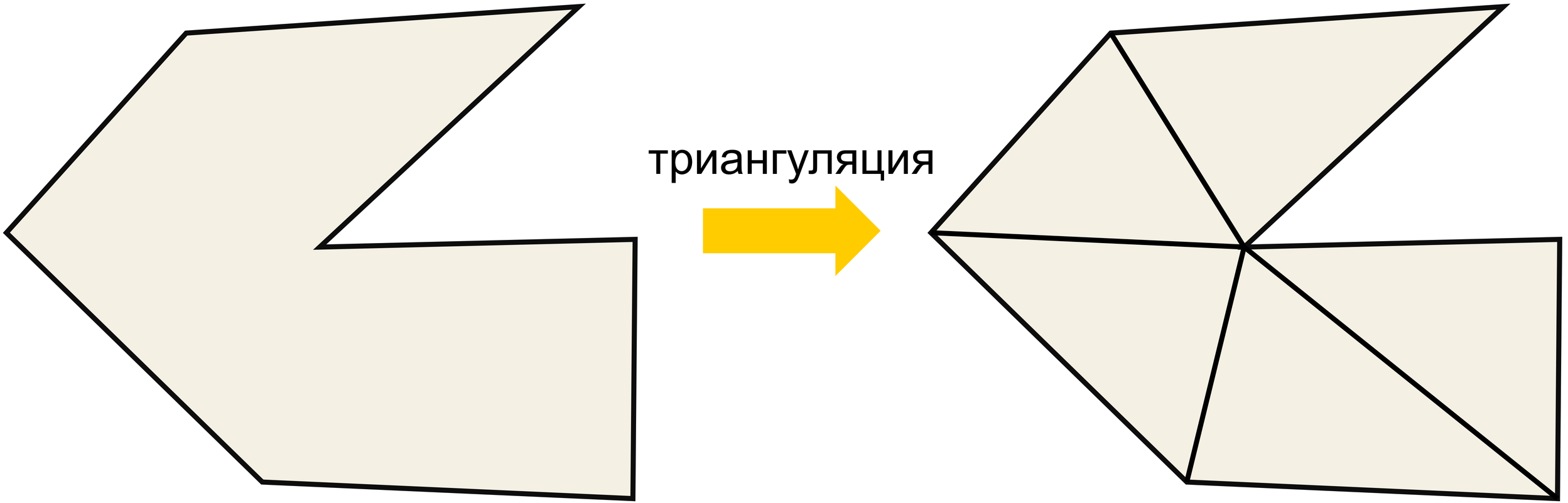
protobuf

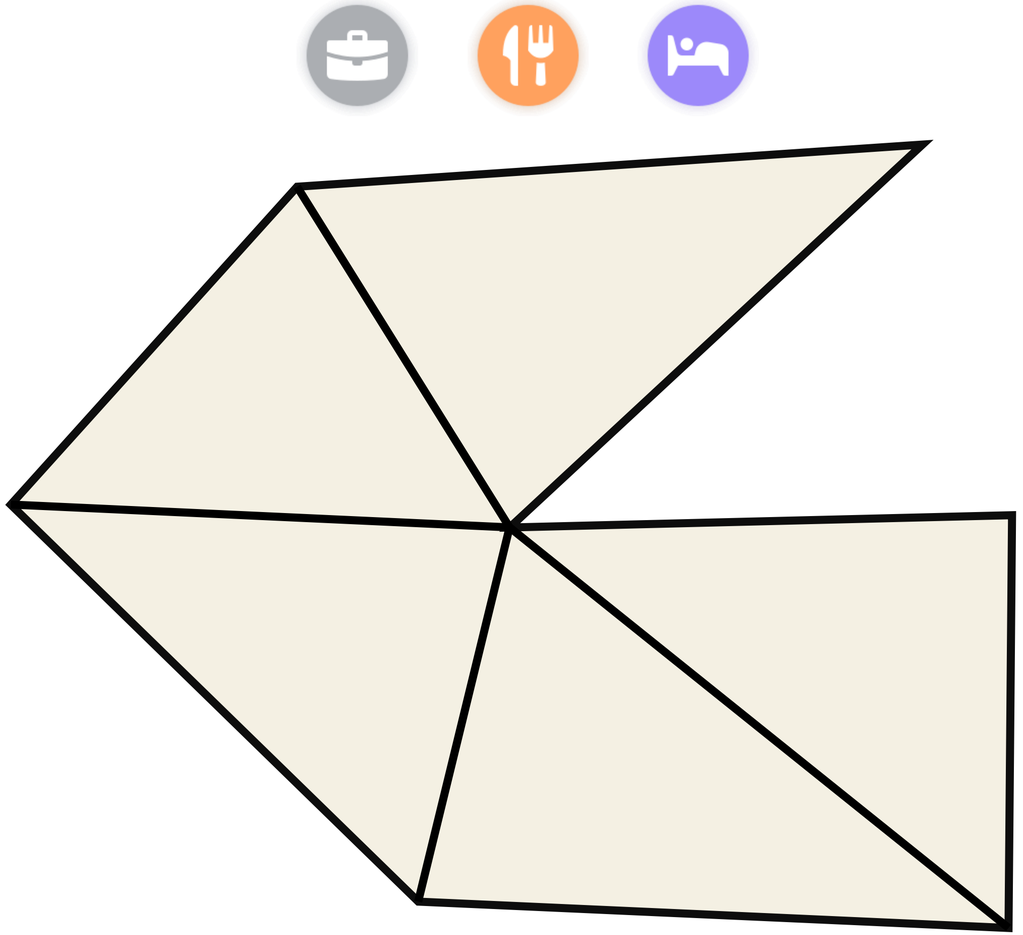
```
1
2  Ä•
3  1••••admin.locality_a.locality_0•••"••••"•""••ÿðùü
4  >•••(urban-area.landuse_a.urban-residential_0•••Ò••••"•""••ÿòúý
5  2••••vegetation.veg_residential_0•••Ê••••"•""••ÿÎðß
6  +••••vegetation.veg_lawn_0•••ô••••"•""••ÿÎðß
7  8•••"urban-area.landuse_a.sportground_0••••••••"•""••ÿÎðß
8  •••••2label.transport.station.bus.transport_p.bus-stop_0•u•äÃÜ••••"•"*f
9  '
10  GBZT3ZZLMSRGRXRPSVESLHMRHB4E6JBA•••••••••?••••?"'
11  YTPHLN4TZ5NKUWO2XDHVW5JAN6GF2PLE•••••-•••?5•••?2•
12 Æ•••••poi.poi_large_0•°••àÅÜ••••"•"*f
13 '
14  MTOTZ2S6I34O22QPK2HVMBBYBZPSLATZ•••••••••?••••?"'
15  BSLZZQ3SL4EWGBXRGP7TMZ3QJNTB6MDH•••••-•••?5•••?2;
16 9
17 ÿ••••(87931be60dd7d1ced983a8030b11bd153c5e45ed••É•@%ÿÿÿÿ
18 Æ•••••poi.poi_large_0•°••àÅÜ••••"•"*f
19 '
20  YBORFAZQNY6ELZR6UK4AWSR54FUFTJFA•••••••••?••••?"'
21  IB3H6EIXZB4THW26QH7RWURJBNCZWDLC•••••-•••?5•••?2;
22 9
23 ÿ••••(87931be60dd7d1ced983a8030b11bd153c5e45ed••É•@%ÿÿÿÿ
24 Æ•••••poi.poi_large_0•°••àÅÜ••••"•"*f
25 '
```

protobuf

```
1
2  Ä•
3  1••••admin.locality_a.locality_0•••¨••••"•""••ÿðùü
4  >•••(urban-area.landuse_a.urban-residential_0•••Ò••••"•""••ÿòúý
5  2••••vegetation.veg_residential_0•••Ê••••"•""••ÿÎðß
6  +••••vegetation.veg_lawn_0•••ô••••"•""••ÿÎðß
7  8•••"urban-area.landuse_a.sportground_0••••••••"•""••ÿÎðß
8  •••••2label.transport.station.bus.transport_p.bus-stop_0•u•äÃÜ••••"•"*f
9  '
10  GBZT3ZZLMSRGRXRPSVESLHMRHB4E6JBA•••••••••?••••?"'
11  YTPHLN4TZ5NKUWO2XDHVW5JAN6GF2PLE•••••-•••?5•••?2•
12 Æ•••••poi.poi_large_0•°••àÅÜ••••"•"*f
13 '
14  MTOTZ2S6I34O22QPK2HVMBBYBZPSLATZ•••••••••?••••?"'
15  BSLZZQ3SL4EWGBXRGP7TMZ3QJNTB6MDH•••••-•••?5•••?2;
16 9
17 ÿ••••(87931be60dd7d1ced983a8030b11bd153c5e45ed••É•@%ÿÿÿÿ
18 Æ•••••poi.poi_large_0•°••àÅÜ••••"•"*f
19 '
20  YBORFAZQNY6ELZR6UK4AWSR54FUFTJFA•••••••••?••••?"'
21  IB3H6EIXZB4THW26QH7RWURJBNCZWDLC•••••-•••?5•••?2;
22 9
23 ÿ••••(87931be60dd7d1ced983a8030b11bd153c5e45ed••É•@%ÿÿÿÿ
24 Æ•••••poi.poi_large_0•°••àÅÜ••••"•"*f
25 '
```

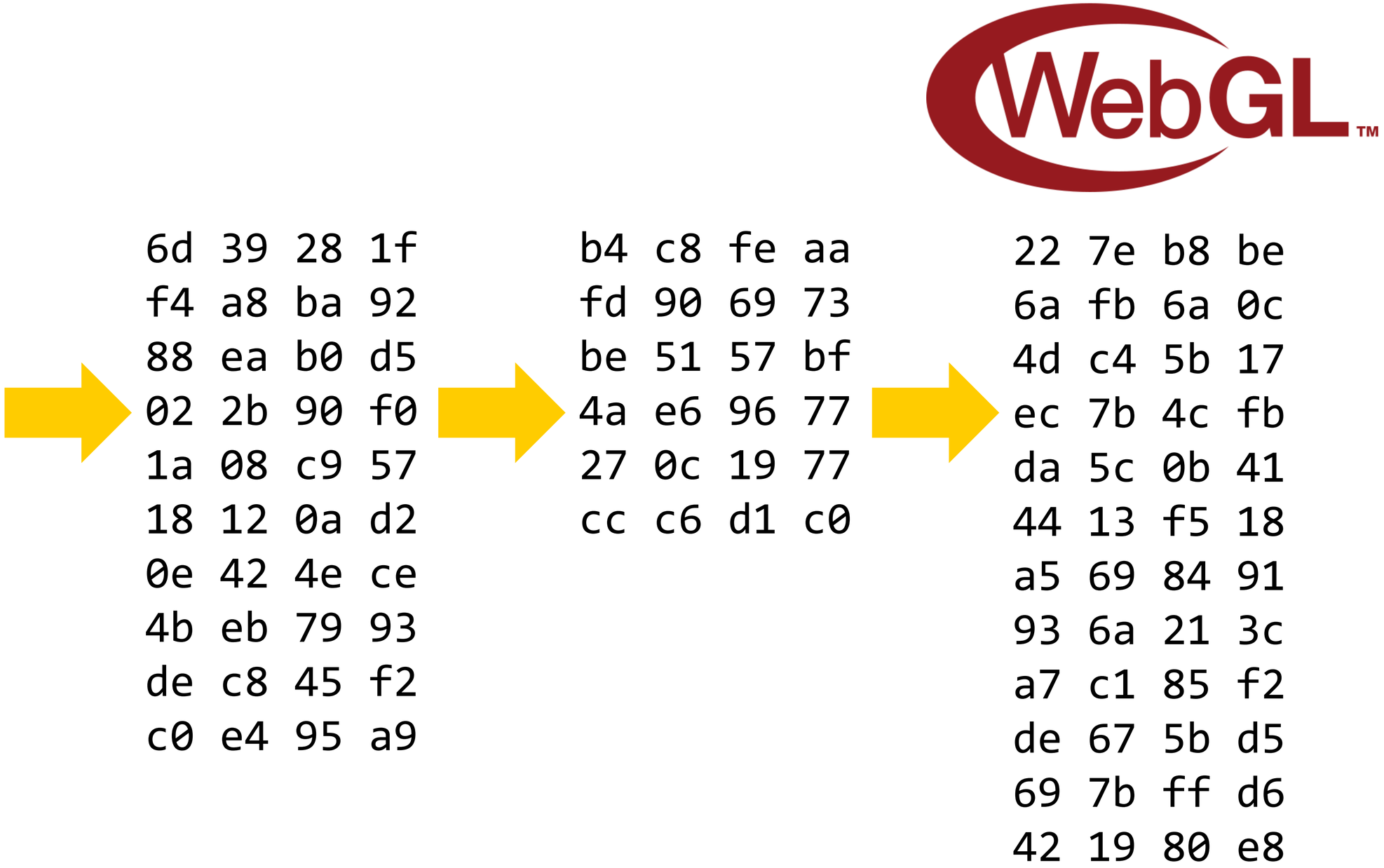



триангуляция

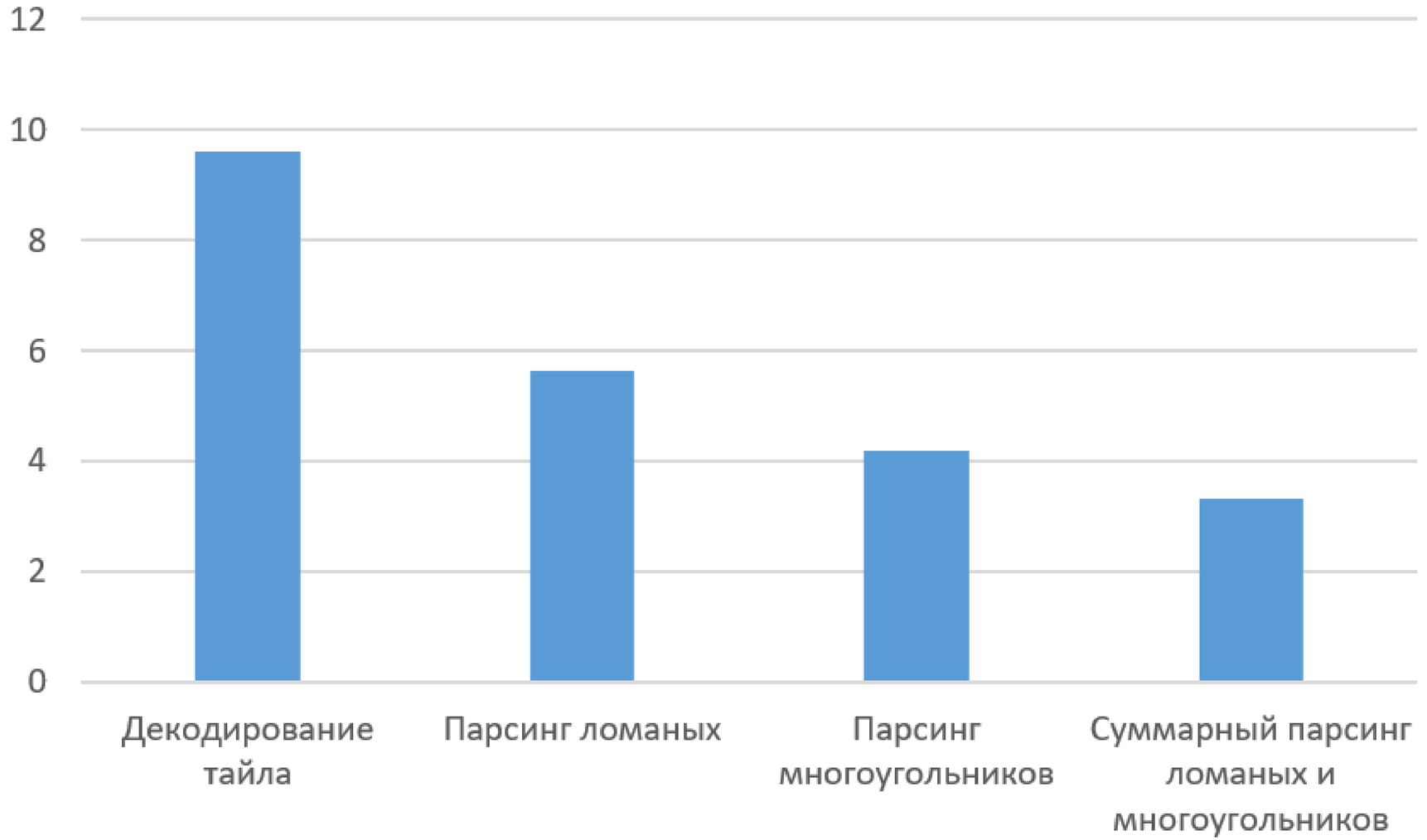



```
88 c6 54 81 b4 4b db 14
81 96 ea 35 66 4c 16 8d
45 65 67 74 65 1e dc 6b
24 d8 a4 6e 59 6b 2d 0e
1a 41 d1 9d b3 7b 91 5f
0f f1 08 f0 5a a7 8d e1
69 57 0a 49 3f 66 a1 8a
35 a8 f4 fb e8 82 db f6
f8 f3 0b 3a 8c 67 0b 5c
b2 71 c6 b1 2d ef 74 60
```

protobuf


bd 47 85 83
13 e3 73 f5
e7 34 f7 55
7a fd 5b 7d
e7 54 6d 40
e3 81 88 cf
52 a3 18 99
fc 05 d7 ea
6d 39 28 1f
f4 a8 ba 92
88 ea b0 d5
02 2b 90 f0
1a 08 c9 57
18 12 0a d2
0e 42 4e ce
4b eb 79 93
de c8 45 f2
c0 e4 95 a9
b4 c8 fe aa
fd 90 69 73
be 51 57 bf
4a e6 96 77
27 0c 19 77
cc c6 d1 c0
22 7e b8 be
6a fb 6a 0c
4d c4 5b 17
ec 7b 4c fb
da 5c 0b 41
44 13 f5 18
a5 69 84 91
93 6a 21 3c
a7 c1 85 f2
de 67 5b d5
69 7b ff d6
42 19 80 e8

emscripten


12
10
8
6
4
2
0
Декодирование тайла
Парсинг ломаных
Парсинг многоугольников
Суммарный парсинг ломаных и многоугольников

Ускорение отдельных операций – 3х

Ускорение всей работы – 20-25%

Минусы

- Параллельная кодовая база на C++
- Параллельный разработчик на C++
- Параллельные баги на C++
- Сильно увеличивается размер бандла
- Не подходит для всего приложения: много дополнительного кода и накладных расходов

Пока мы так и не решились полностью перевести часть кода на WebAssembly

Подробнее

- Скоро будет статья в блоге Яндекса на хабре
- Moscow JS Geo Meetup. Как мы рендерим векторные Яндекс.Карты на вебе

Мы скачали emscripten.
Написали код на C++.
Что-то там скомпилировалось.
На выходе получился какой-то бинарник.
Он как-то запускается в браузере.
Почему-то работает быстро.


WEBASSEMBLY

WebAssembly

Что это такое

- Portable, size- and load-time-efficient binary format
- Compilation target
- Can be executed at native speed by taking advantage of common hardware capabilities
- Roughly the same functionality as asm.js
- Primarily aimed at C/C++
- Design to execute within and integrate well with the existing Web platform
- Maintain the versionless, feature-tested and backwards-compatible evolution story of the Web

https://webassembly.org/docs/high-level-goals/


WA

Asm.js

WA

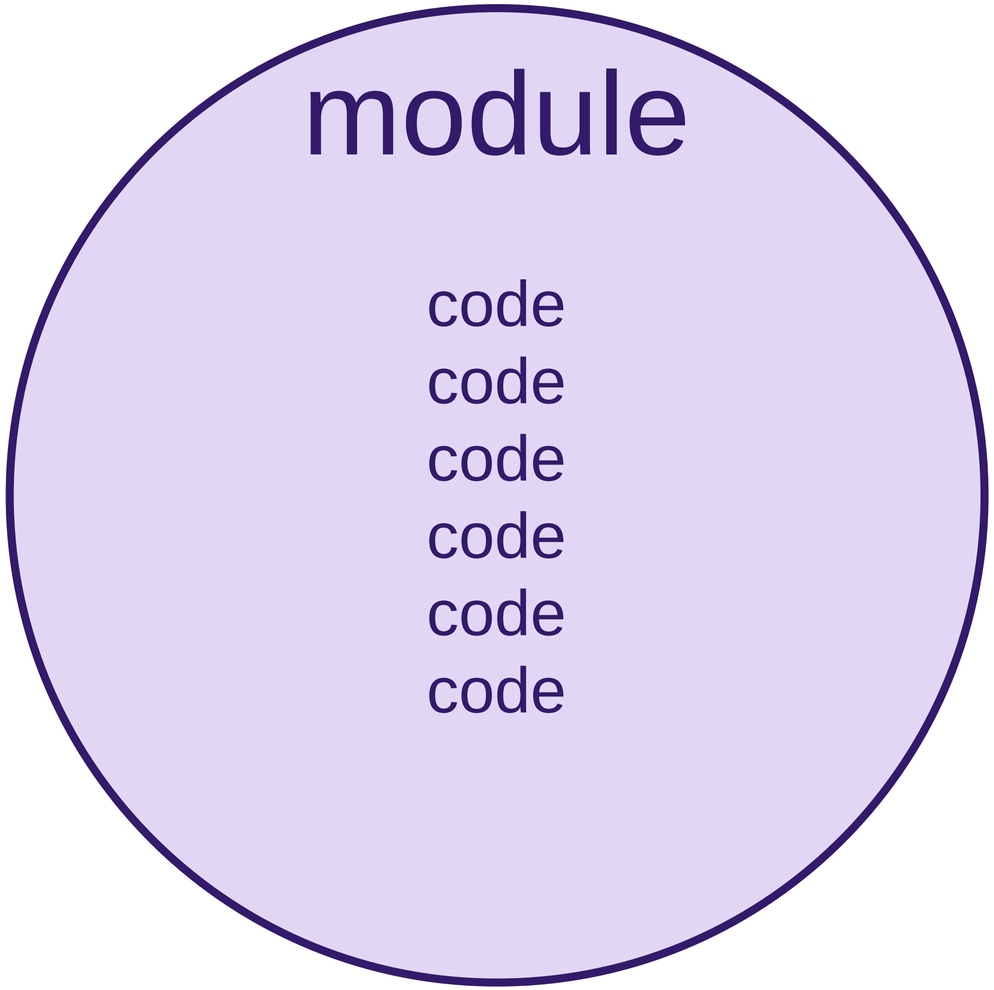
NaCl

Зачем?

	.net jvm flash js	wasm	x86/arm/… nacl
Абстрагирован от конкретного железа?	да	да	нет
Уровень абстракции	высокий	средний	нулевой
Скорость	низкая	средняя	высокая
Система типов	сложная	минималистичная	каша из байтов для команд и данных без обязательной структуры
Структурированность	да	да	
Сложность	сложно	не очень сложно	
Сборка мусора	да	нет	
Дополнительный софт	да	нет	

Как устроен WebAssembly?

Модуль



Типы данных

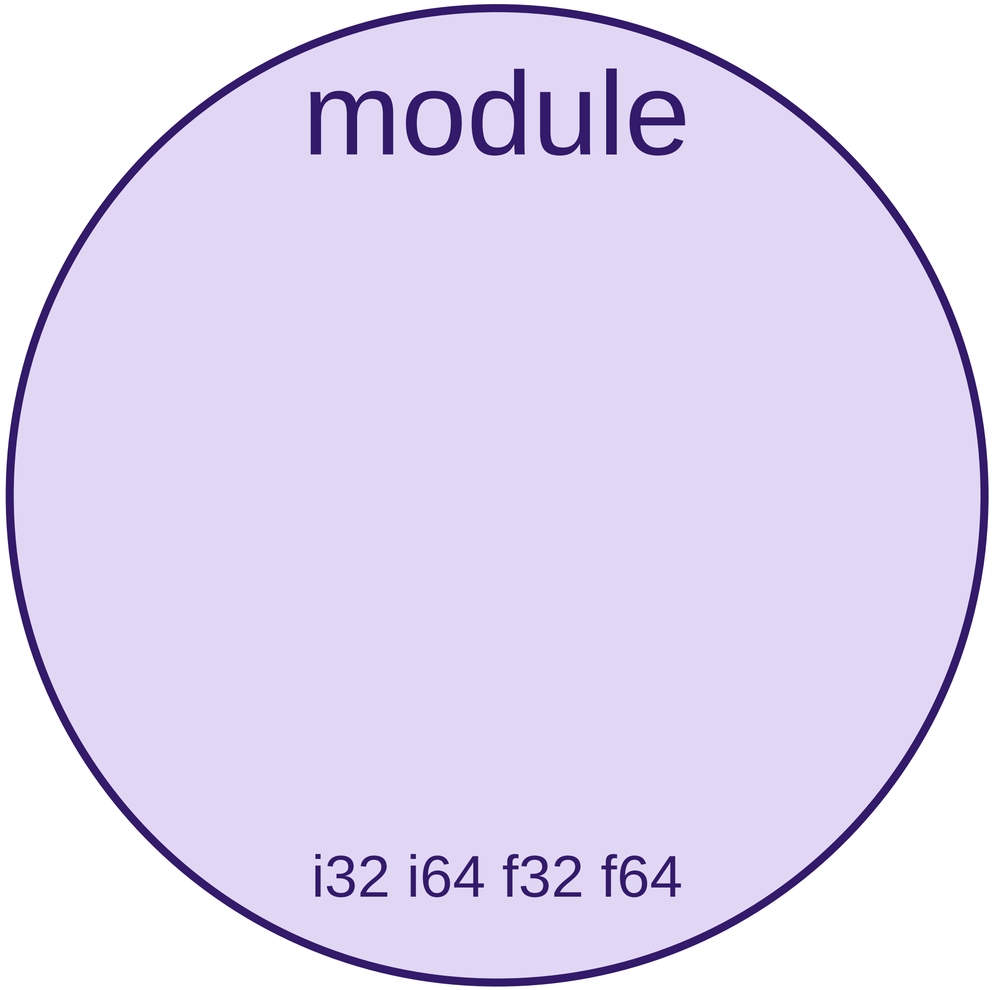
- `f32`, `f64` – числа с плавающей запятой
  - IEEE754
  - По-сути, то же, что и `number` в JavaScript

- `i32`, `i64` – целые числа
  - Отрицательные числа представляются через дополнительный код

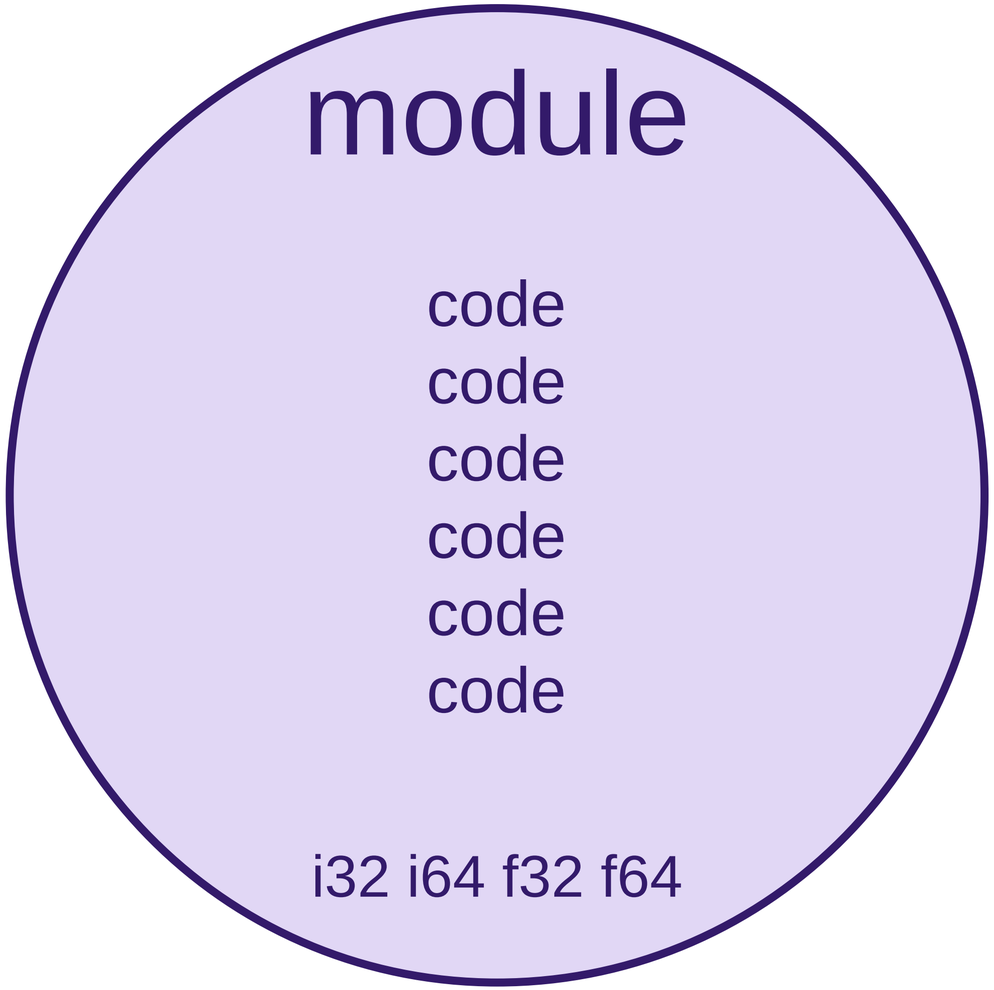
…

- Больше ничего нет
  - Вместо `boolean` – 0 и 1

Типы данных



Инструкции



Инструкции

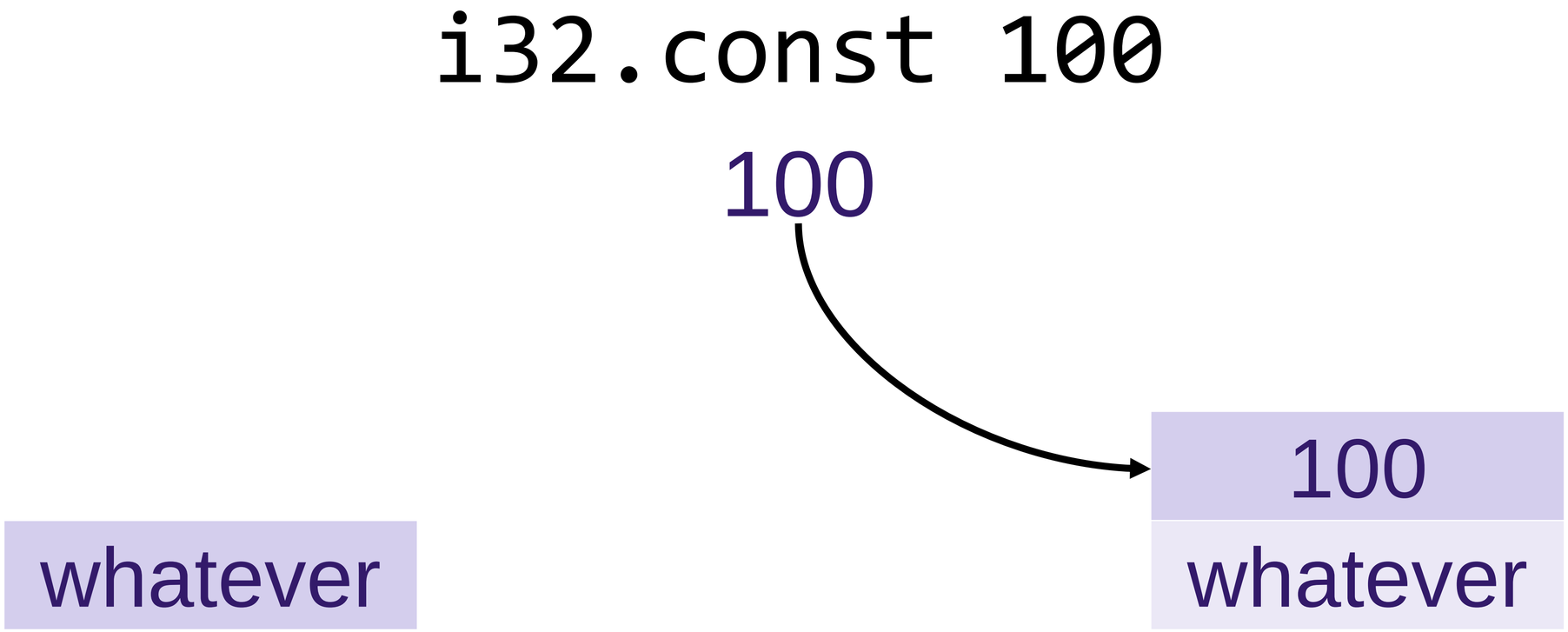
`type.operation`

Обычная математика: add, sub, mul, div

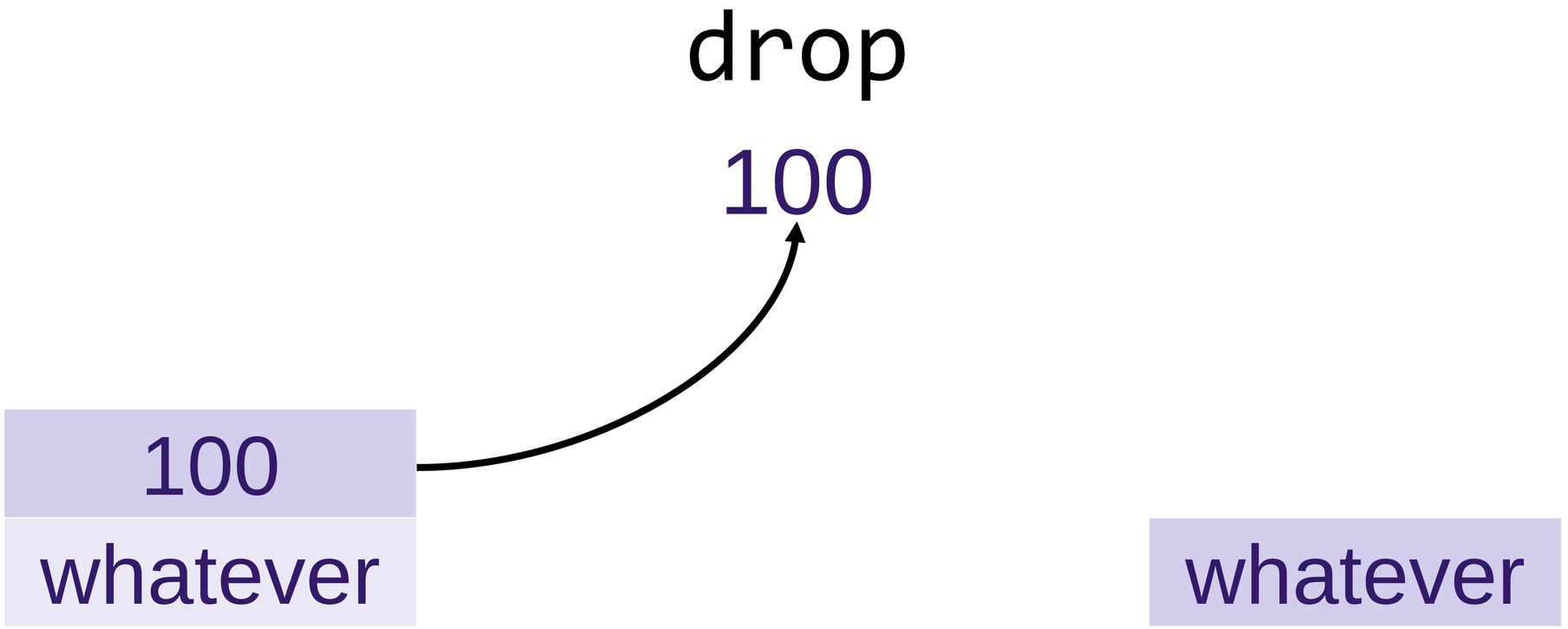
Битовые операции: and (&), or (|), xor (^), shl/shr (>>/<<), …

Операции сравнения: eq (==), ne (!=), eqz (==0), gt/lt (>/<), ge/le (>=/<=), …

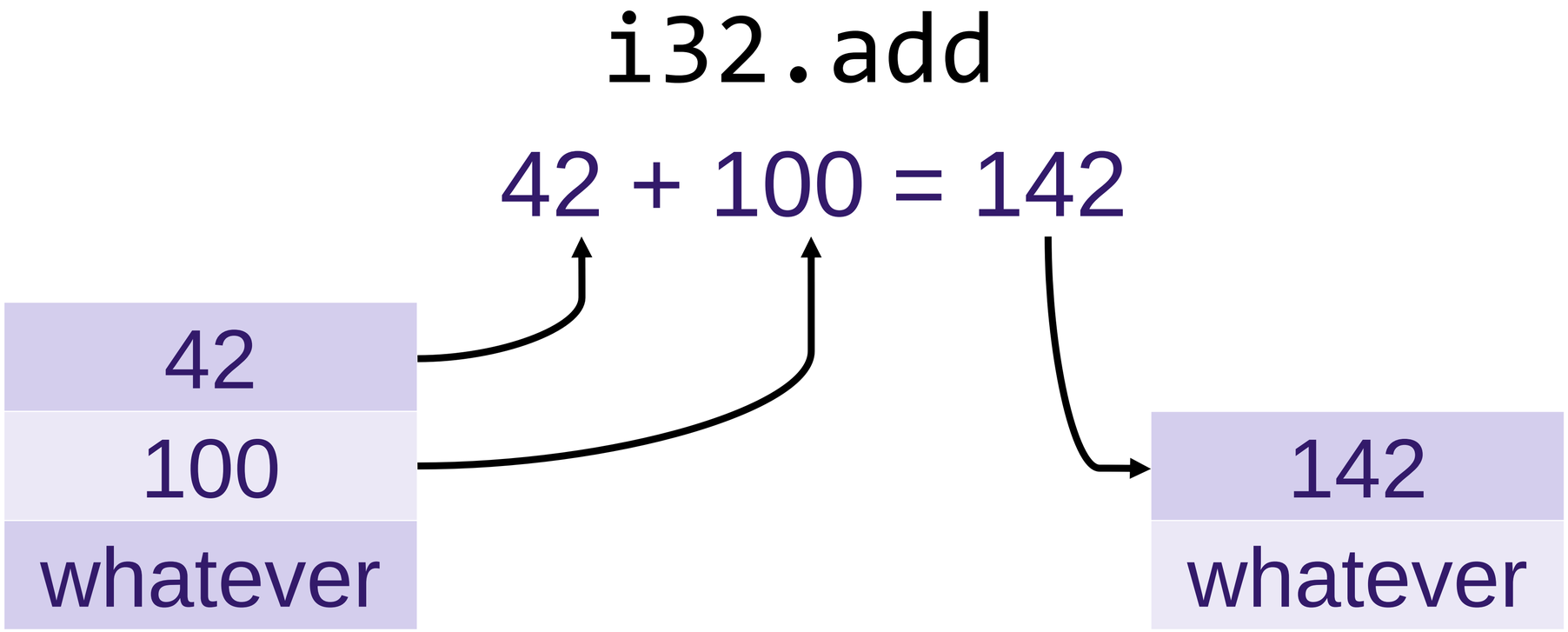
Стековая машина



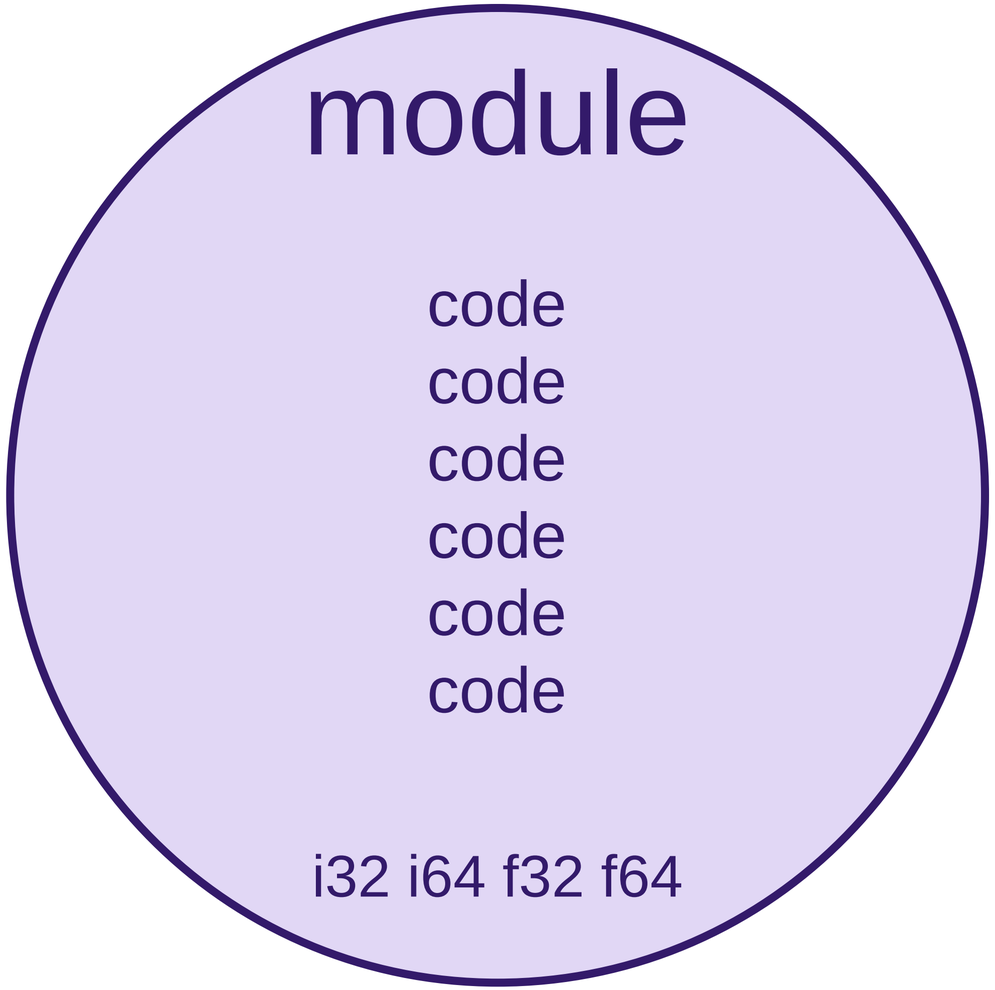
Стековая машина



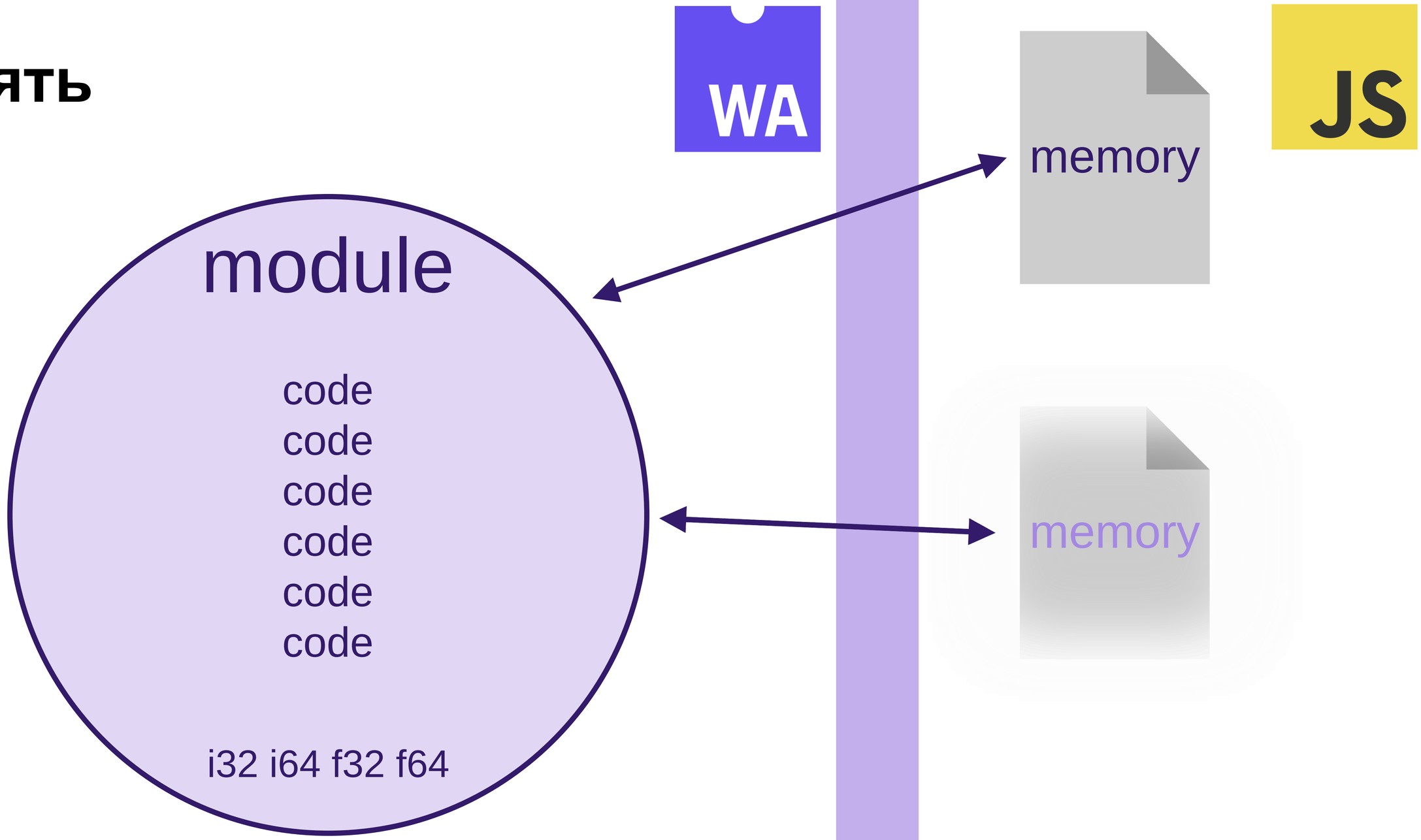
Стековая машина



Интеграция



Память


WA
JS
memory
module
code
code
code
code
code
code
i32 i64 f32 f64
memory

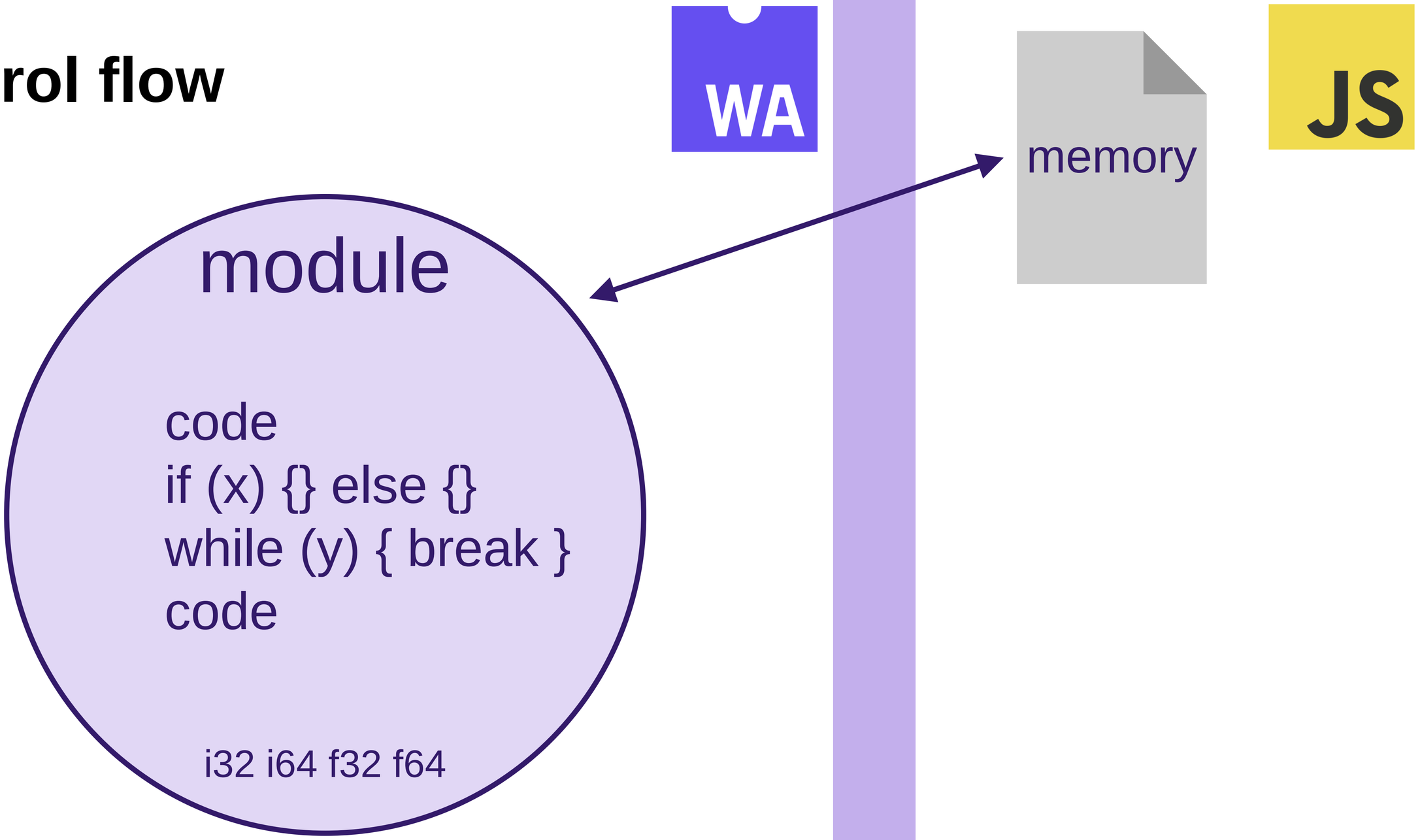
Память

```
const memory = new ArrayBuffer(n * 65536);
```

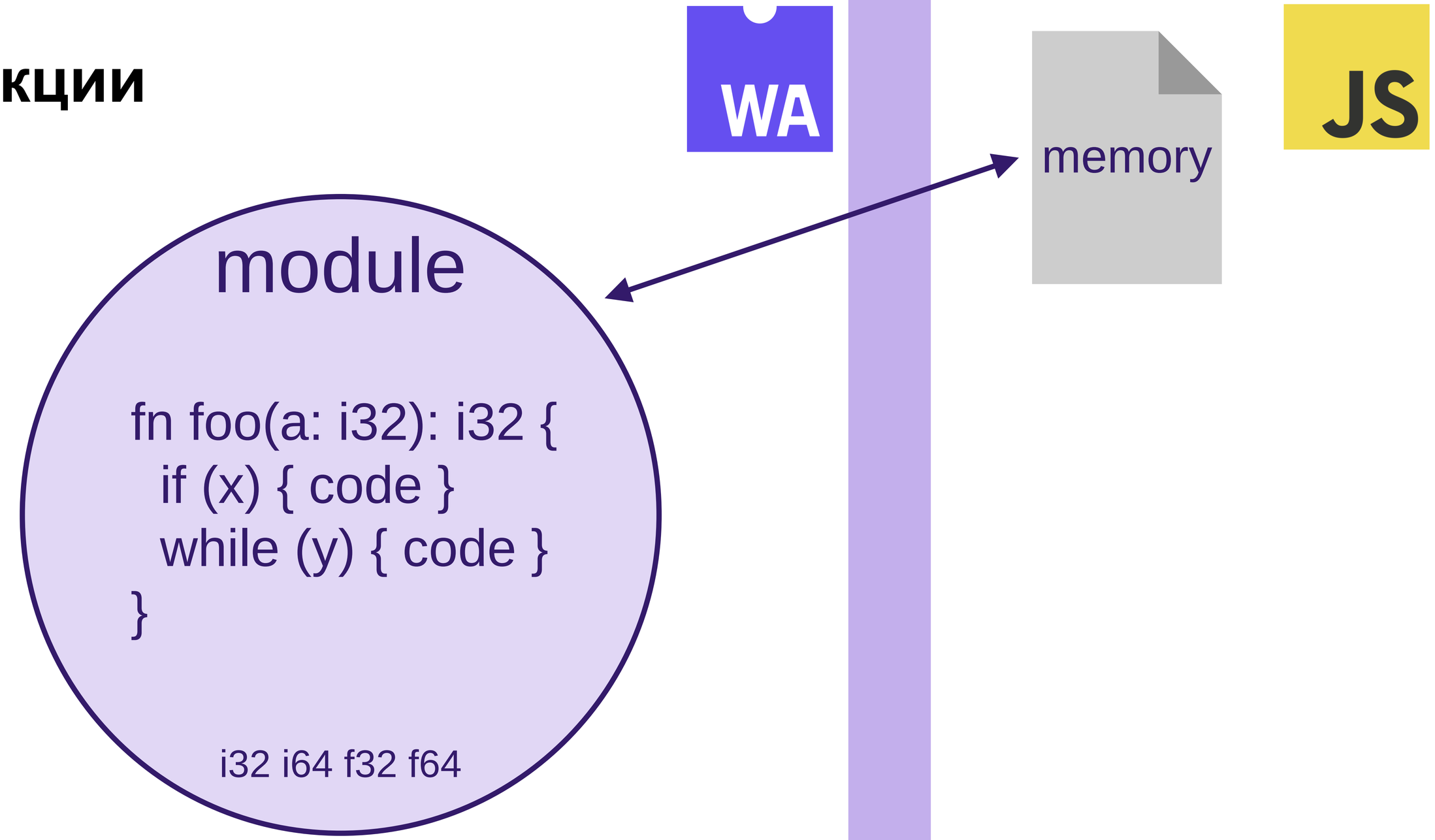
- Можно читать
  - `iXX.load – dataView.getIntXX()`
  - `fXX.load – dataView.getFloatXX()`
- Можно писать
  - `iXX.store – dataView.setIntXX()`
  - `fXX.store – dataView.setFloatXX()`

- Память - little endian
  - `0xaabbccdd => dd cc bb aa`

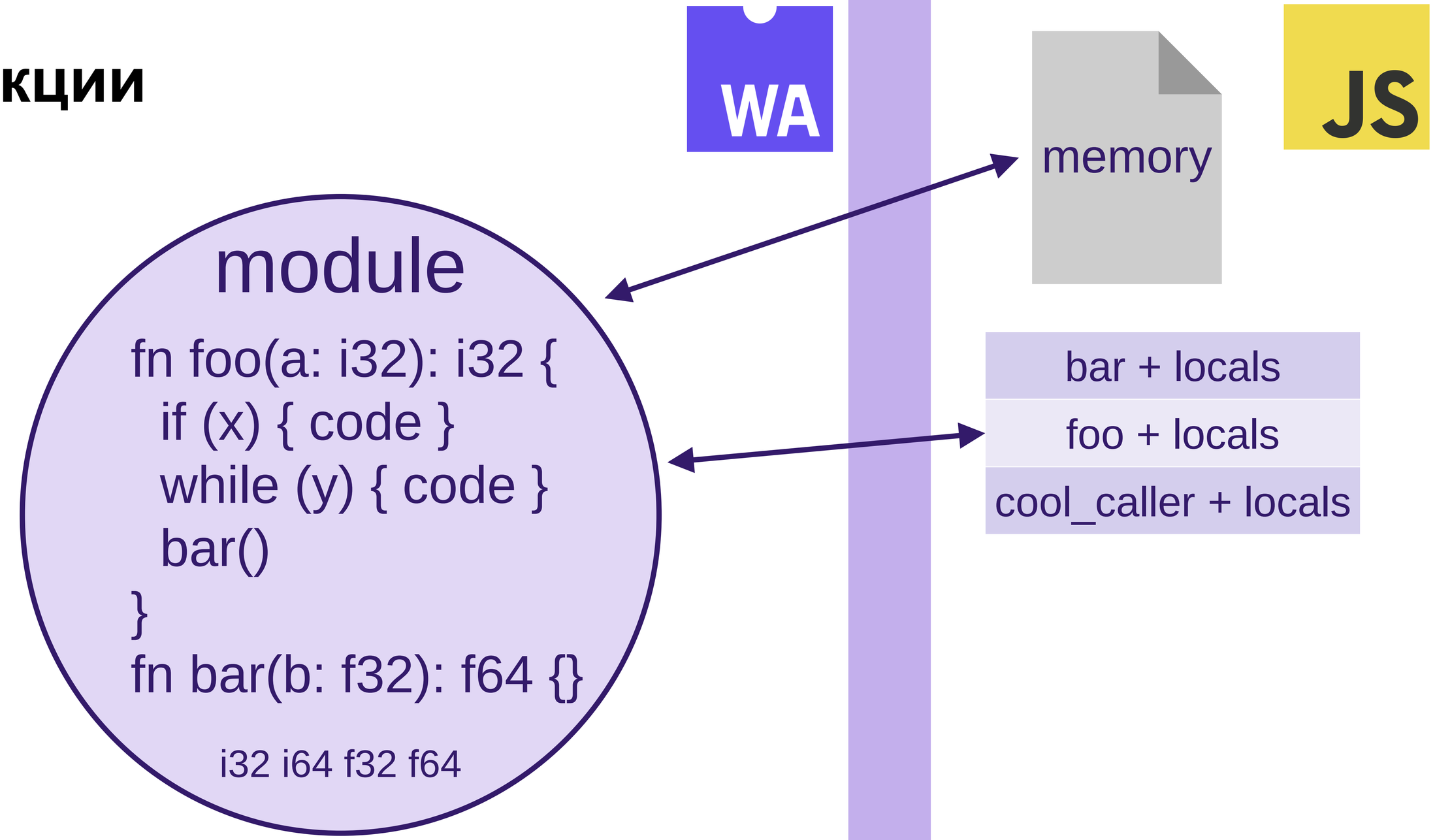
Control flow



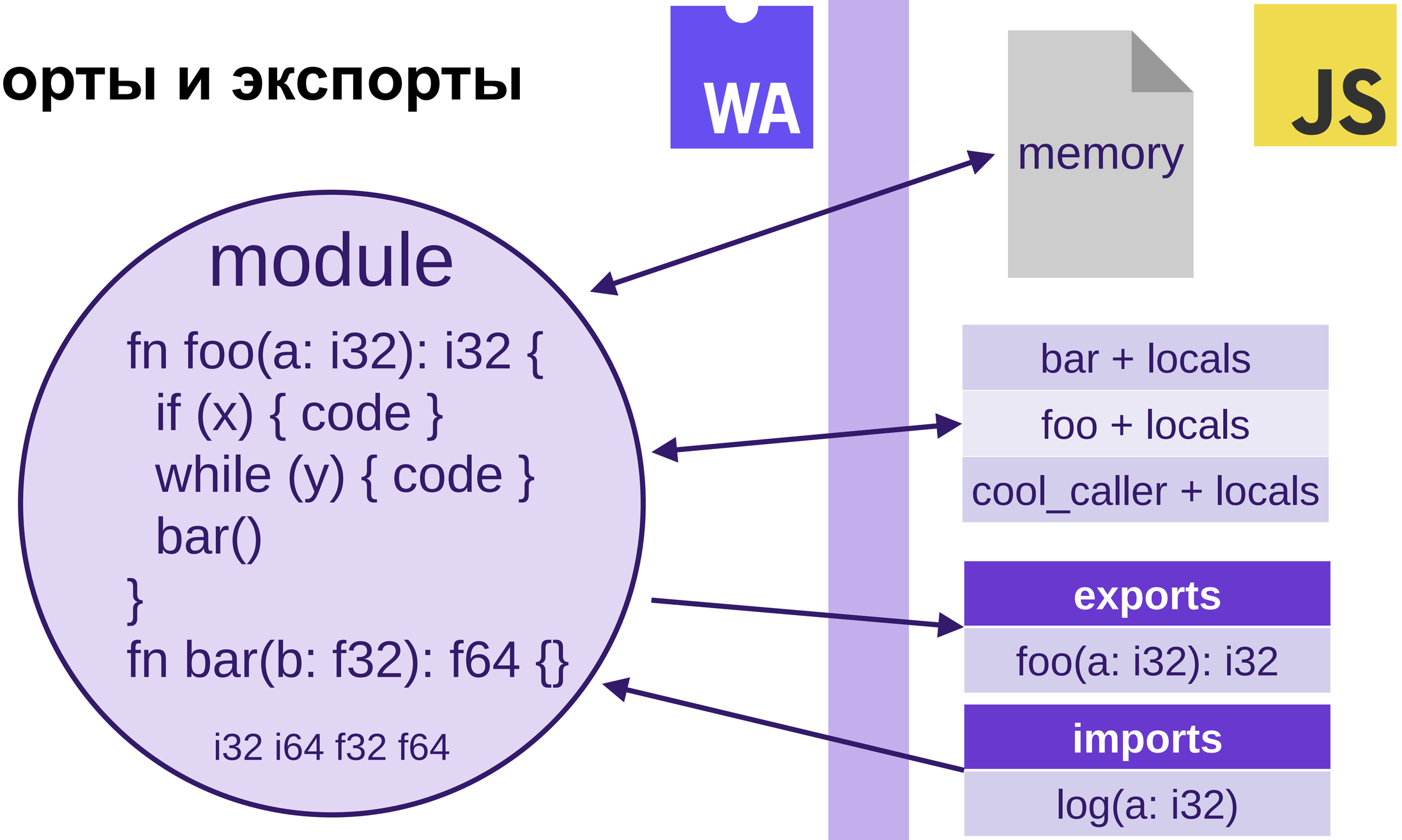
Функции


WA
memory
JS
module
fn foo(a: i32): i32 {
if (x) { code }
while (y) { code }
}
i32 i64 f32 f64

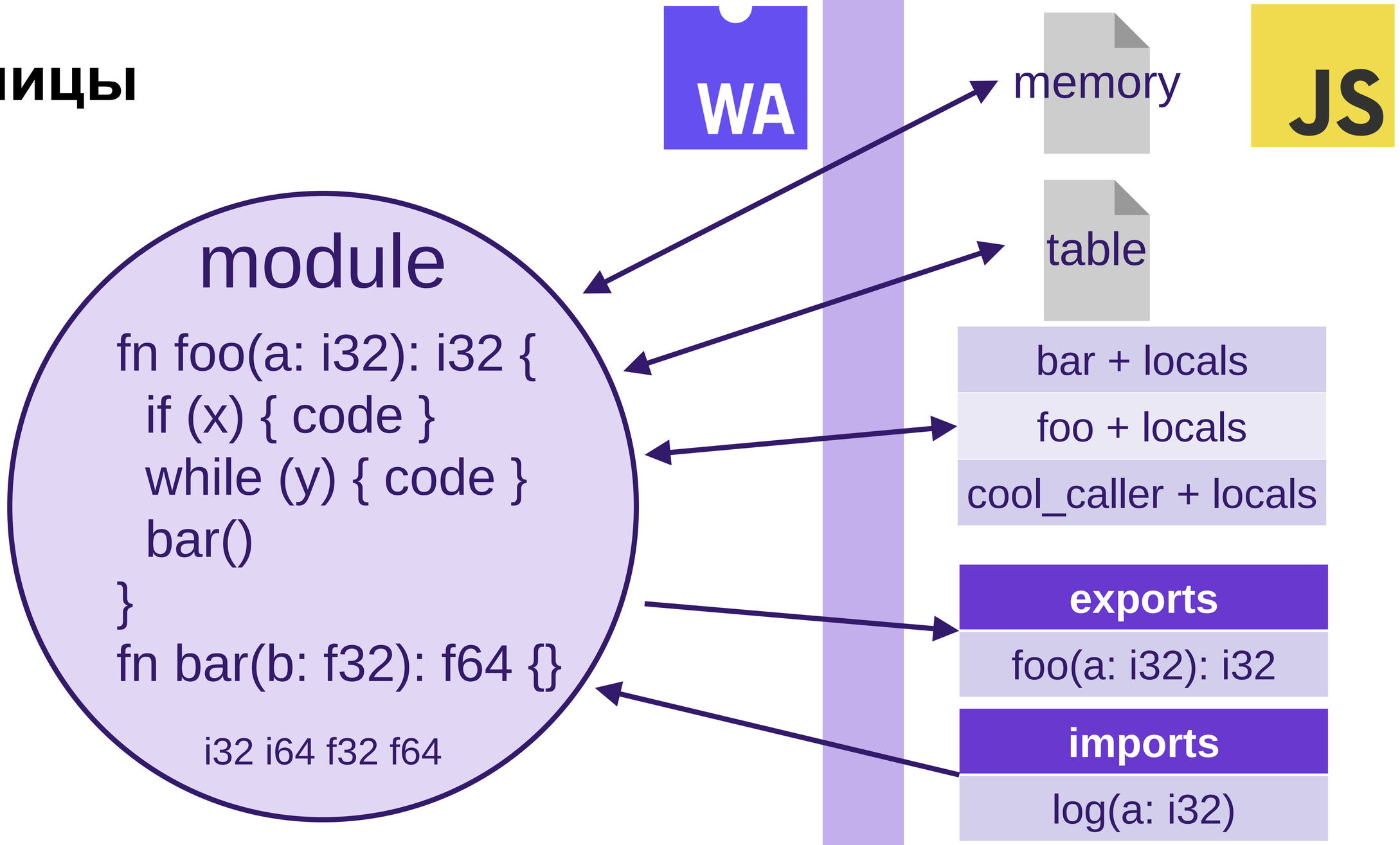
Функции


WA
JS
memory
module
fn foo(a: i32): i32 {
if (x) { code }
while (y) { code }
bar()
}
fn bar(b: f32): f64 {}
i32 i64 f32 f64
bar + locals
foo + locals
cool_caller + locals

Импорты и экспорты



Таблицы


WA
JS
memory
table
module
fn foo(a: i32): i32 {
if (x) { code }
while (y) { code }
bar()
}
fn bar(b: f32): f64 {}
i32 i64 f32 f64
bar + locals
foo + locals
cool_caller + locals
exports
foo(a: i32): i32
imports
log(a: i32)


That's all Folks!
WA

Как этого хватает?

JavaScript модули

```
export function fancyButton({text, onClick}) {
    const handler = () => {
        console.log('clicked', Date.now());
        onClick();
    };

    return <button onClick={handler}>{text}</button>;
}
```

JavaScript модули

```
export function fancyButton({text, onClick}) {
    const handler = () => {
        console.log('clicked', Date.now());
        onClick();
    };

    return <button onClick={handler}>{text}</button>;
}
```

JavaScript модули

```
export function fancyButton({text, onClick}) {
    const handler = () => {
        console.log('clicked', Date.now());
        onClick();
    };

    return <button onClick={handler}>{text}</button>;
}
```

JavaScript модули

```
export function fancyButton({text, onClick}) {
    const handler = () => {
        console.log('clicked', Date.now());
        onClick();
    };

    return <button onClick={handler}>{text}</button>;
}
```

JavaScript модули

```
import * as React from 'react';

export function fancyButton({text, onClick}) {
    const handler = () => {
        console.log('clicked', Date.now());
        onClick();
    };

    return React.createElement('button', {onClick: handler}, text);
}
```

JavaScript модули

```
import * as React from 'react';

export function fancyButton({text, onClick}) {
    const handler = () => {
        console.log('clicked', Date.now());
        onClick();
    };

    return React.createElement('button', {onClick: handler}, text);
}
```

JavaScript модули

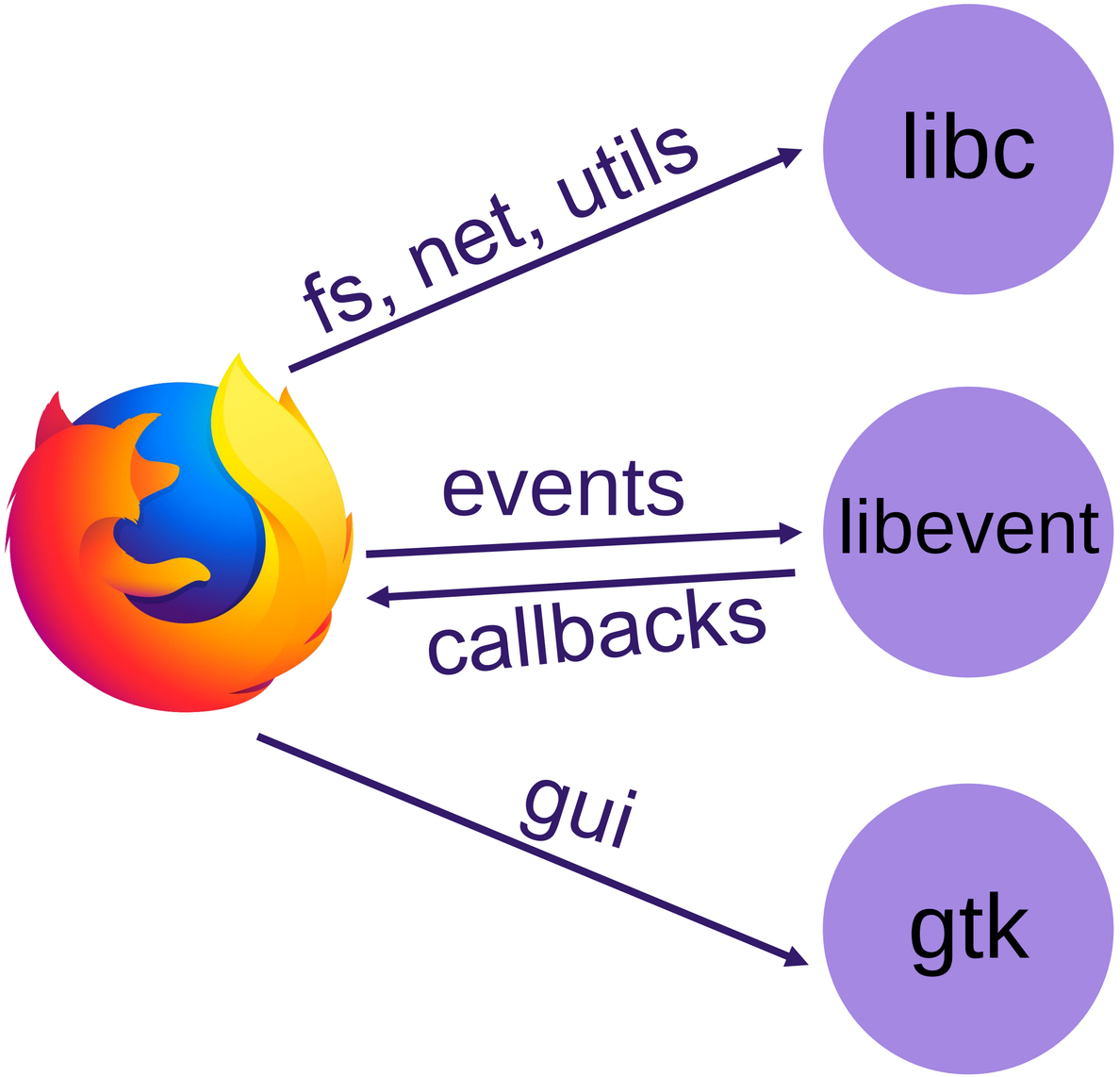
```
import * as React from 'react';
import * as Date from 'javascript-standard-library';

export function fancyButton({text, onClick}) {
    const handler = () => {
        console.log('clicked', Date.now());
        onClick();
    };

    return React.createElement('button', {onClick: handler}, text);
}
```

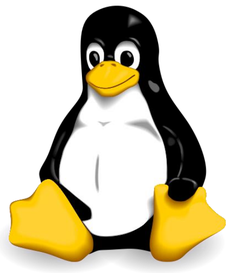
JavaScript модули

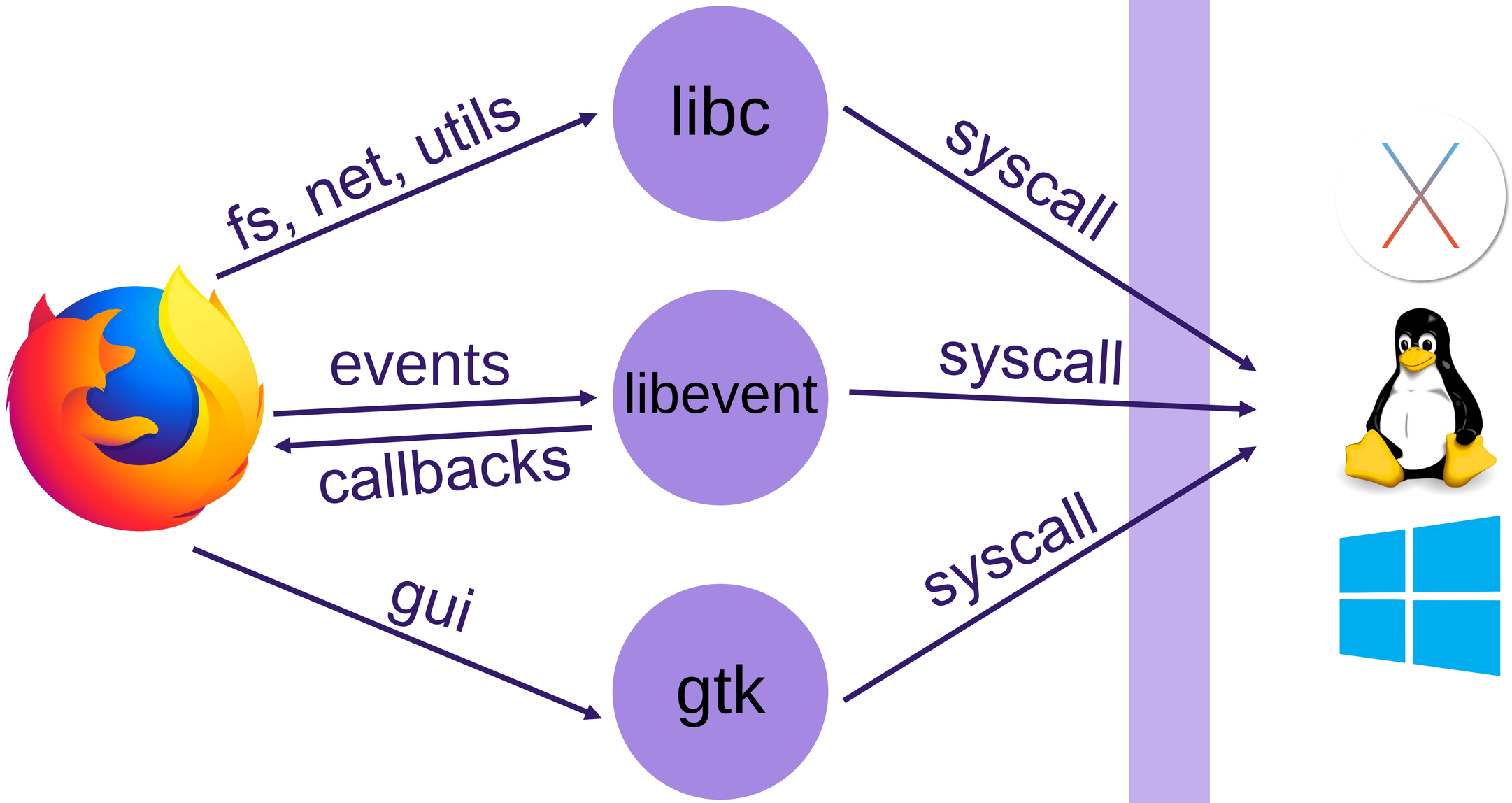
```
import * as React from 'react';
import * as Date from 'javascript-standard-library';

export function fancyButton({text, onClick}) {
    const handler = () => {
        console.log('clicked', Date.now());
        onClick();
    };

    return React.createElement('button', {onClick: handler}, text);
}
```

JavaScript модули

```
import * as React from 'react';
import * as Date from 'javascript-standard-library';

export function fancyButton({text, onClick}) {
    const handler = () => {
        console.log('clicked', Date.now());
        onClick();
    };

    return React.createElement('button', {onClick: handler}, text);
}
```


libc
fs, net, utils
events
libevent
callbacks
gui
gtk

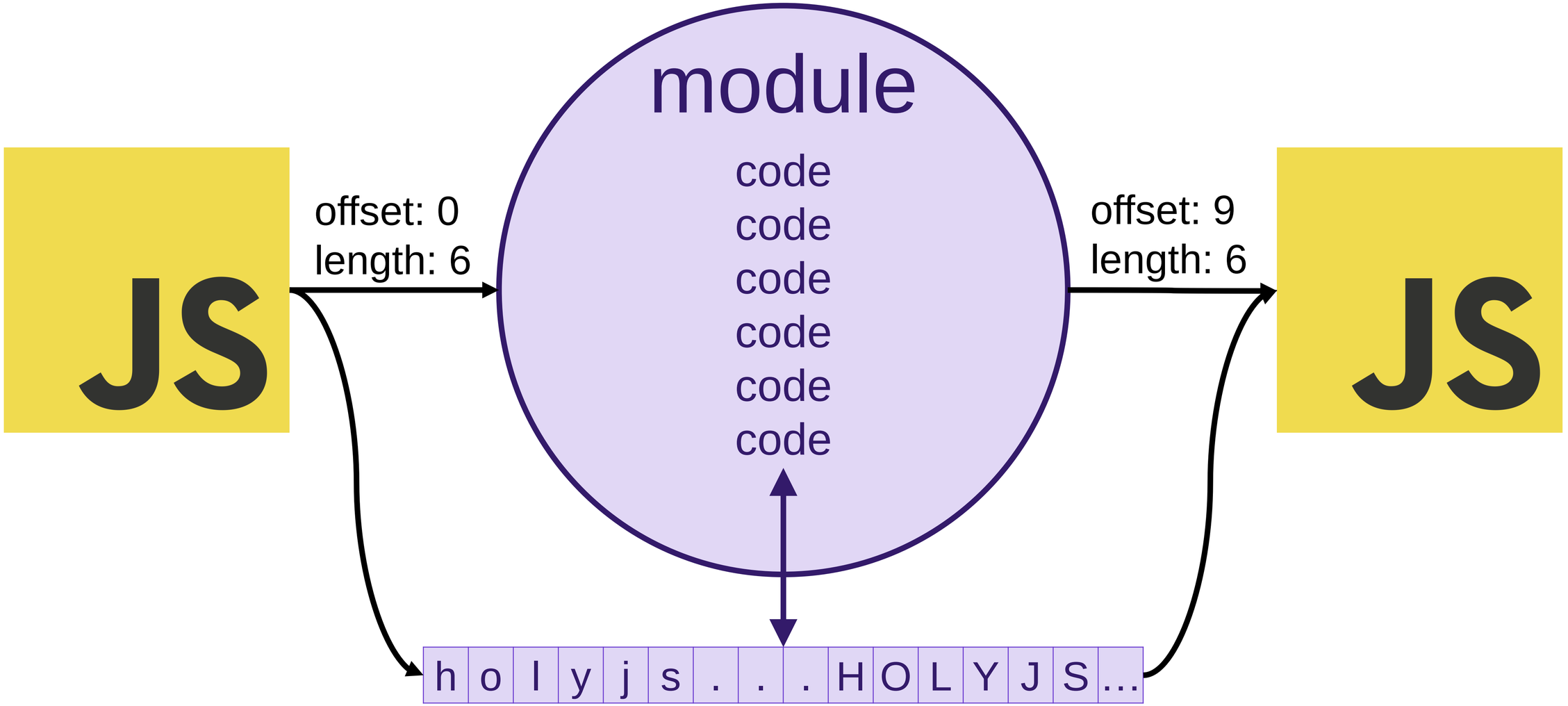



libc
fs, net, utils
syscall
events
libevent
callbacks
syscall
gui
gtk
syscall

Программируем на WebAssembly

Вы скорее всего не будете писать WebAssembly руками

“Сложные” типы



Бинарный формат

```
if   (result i32)
0x04 0x7F
i32.const 42
0x41       0x2A
i32.const 255
0x41       0x01 0xff
i32.add
0x6A
end
0x0B
```

https://webassembly.github.io/spec/core/binary/instructions.html

Бинарный формат

```
(module ;; "\0asm" version
0x00 0x61 0x73 0x6D 0x01 0x00 0x00 0x00
  ;; type section length = 5 bytes
  0x01 0x05
  (type $type0 (func  (param i32)    )
  0x02          0x60  0x01  0x7F  0x00
  ;; import section length = 13 bytes
  0x02 0x0D
  ...
)
```

https://webassembly.github.io/spec/core/binary/modules.html

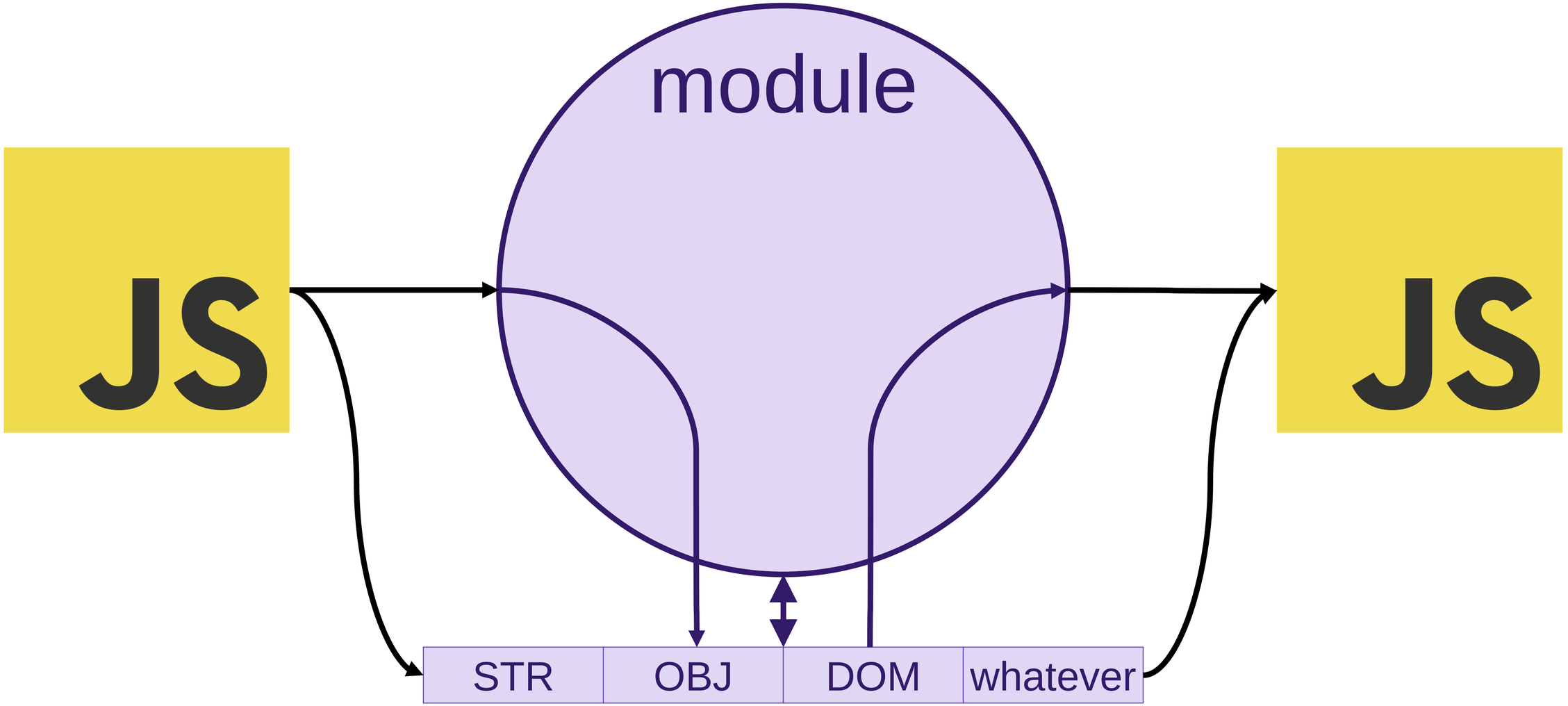
Будущее

Версионирование

- WebAssembly 1.0

- Расширения и feature-detection
  - `try { compile(); supports = run() === expected; } catch { supports = false; }`
  - Возможно будет в самом wasm файле WebAssembly/design#1173

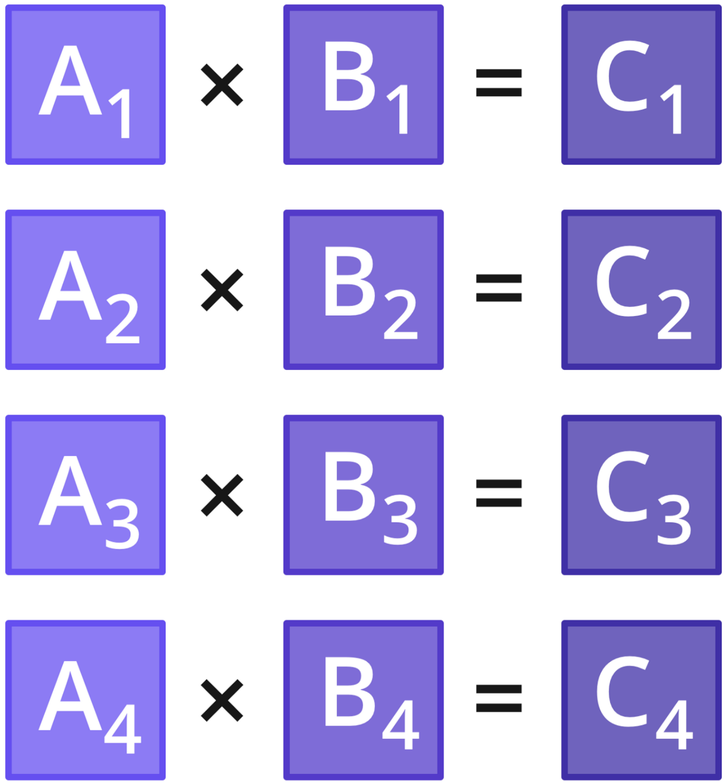
Reference types (anyref)



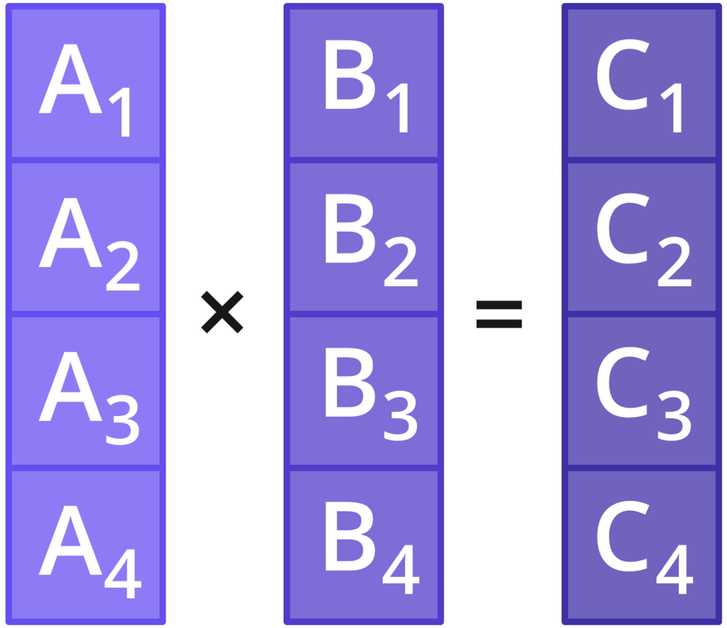
https://hacks.mozilla.org/2019/08/webassembly-interface-types/
https://github.com/WebAssembly/reference-types

SIMD

Scalar Operation



SIMD Operation



https://medium.com/wasmer/webassembly-and-simd-13badb9bf1a8
https://github.com/WebAssembly/simd/blob/master/proposals/simd/SIMD.md

И это еще не всё

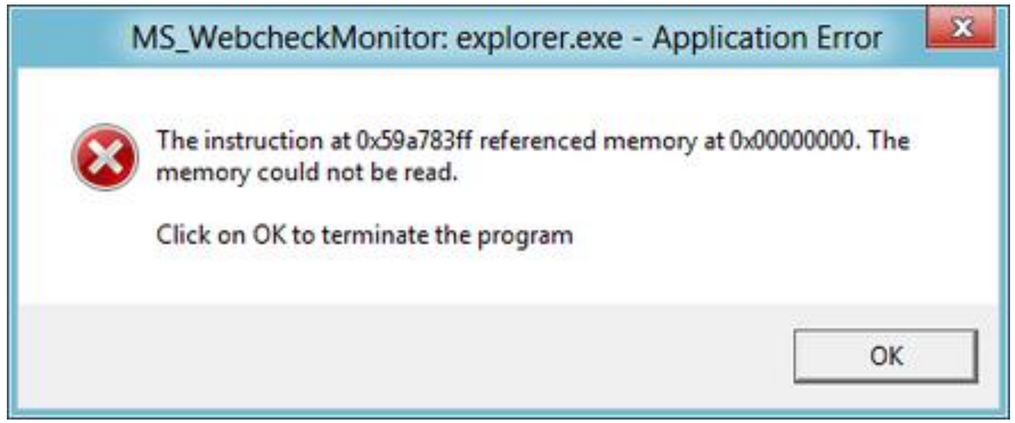
- Exceptions WebAssembly/exception-handling
- Bulk operations (`memset`, `memcpy`) WebAssembly/bulk-memory-operations
- Threads (shared memory, atomics) WebAssembly/threads
- Garbage Collection WebAssembly/gc
- …
- И еще куча всего WebAssembly/proposals

FAQ

Про всякое

- У меня IE11, есть полифил?
- Можно скомпилировать JavaScript в wasm и сделать сайт быстрее?
- Можно внутри wasm скомпилировать wasm и сделать JIT?
- Есть ли string pool, кеш модулей и еще что-нибудь из jvm/.net?

Про безопасность



- Можно ли сделать SEGFAULT?
- Можно ли перезаписать адрес возврата?
- Можно ли перезаписать код?
- Можно ли получить доступ к файловой системе или сети?
- Aaaaa, blazor грузит dll – это новый activex!

Про безопасность

> Худшее зло, которое вы можете причинить, – нагреть комнату своим неэффективным кодом.

Про языки

- C/C++: emscripten, LLVM
- Rust: rustc + bindgen, LLVM
- C#: Mono, Blazor
- Go
- AssemblyScript

Начиная с LLVM 8.0 wasm перестал быть experimental target.

https://github.com/appcypher/awesome-wasm-langs

Не только web

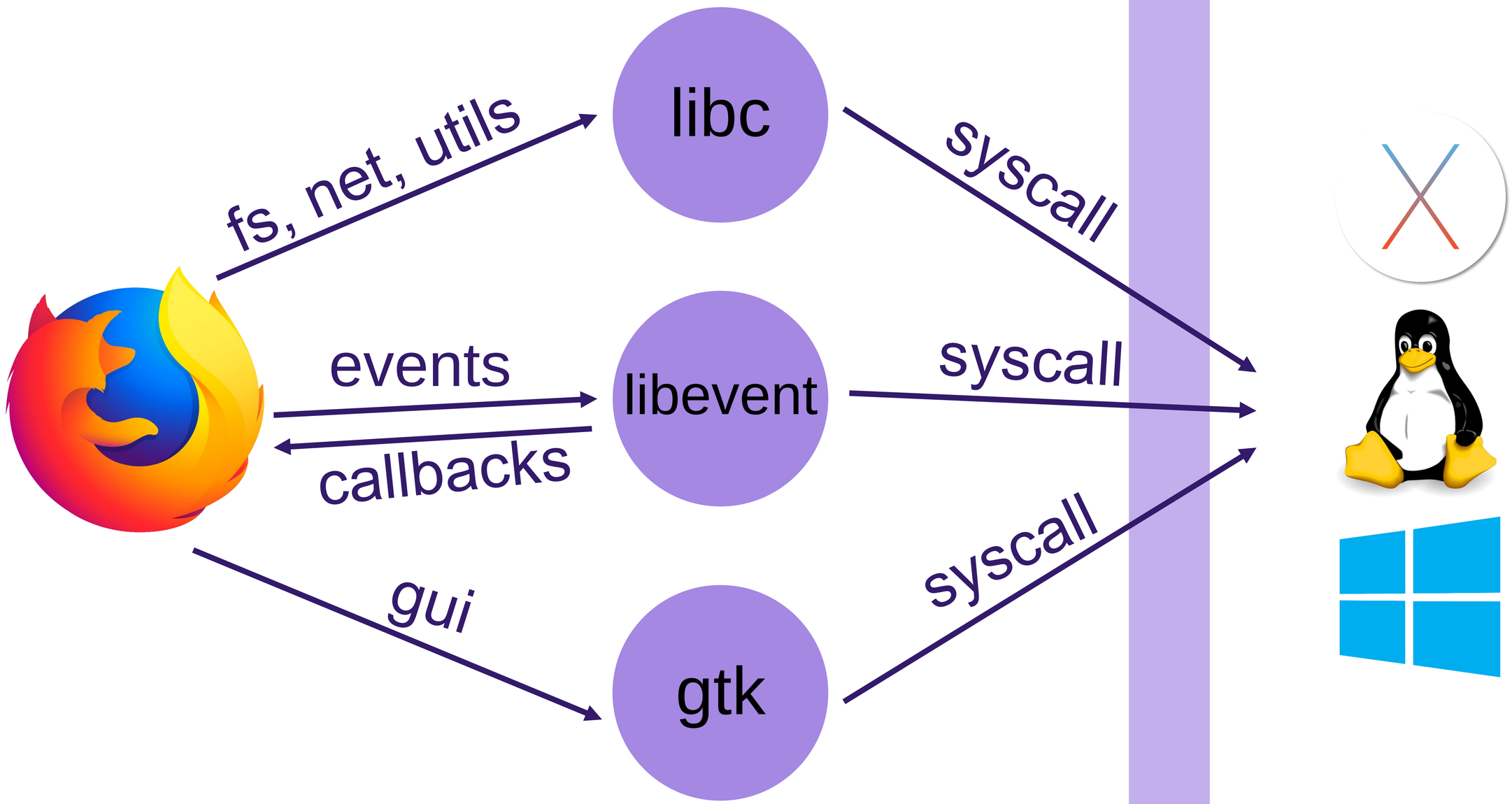
Текущие реализации

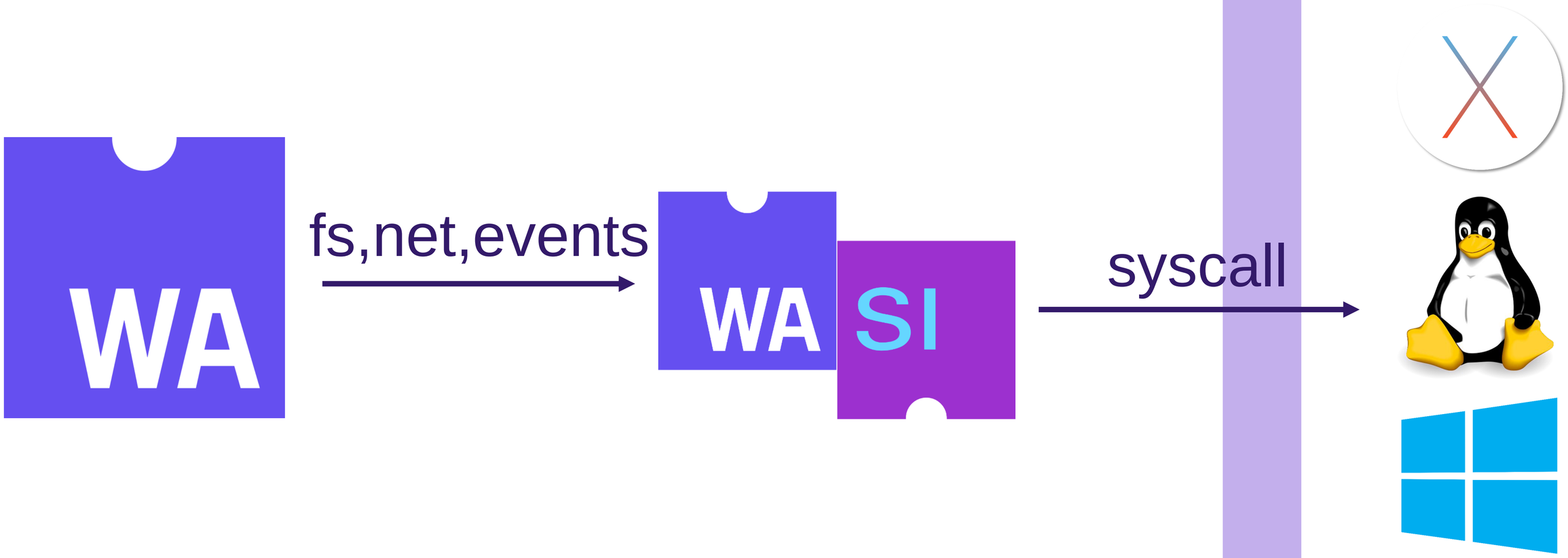
Браузерные движки:

- v8
- SpiderMonkey
- ChakraCore
- JavaScriptCore

Самостоятельные движки:

- binaryen (interpreter + js polyfill + tools)
- wasmer (+wasi)
- wasmtime (wasm meetup #2)


libc
fs, net, utils
syscall
events
libevent
callbacks
syscall
gui
gtk
syscall


WA
fs,net,events
WA
SI
syscall

Заключение

Где вы можете применить это у себя

- Портировать библиотеку и софт
- Числодробилки: фото/видео редакторы, игры
- Использовать один код на бэке и фронте
- Перенос экспертизы ваших коллег из соседней области (blazor)

- Легкая виртуальная машина для различных целей, например, для блокчейнов (wasm meetup #1)
- Figma делала js плагины собирая Duktape и QuickJS в wasm https://www.figma.com/blog/how-we-built-the-figma-plugin-system/

- https://webassembly.org/docs/use-cases/

Полезные инструменты

- WasmExplorer
- WasmFiddle
- WasmCodeExplorer
- WebAssembly Studio

Что почитать

- ttps://webassembly.github.io/spec/core/index.html
- https://hacks.mozilla.org/category/webassembly/

- WebAssembly — русскоговорящее сообщество https://t.me/WebAssembly_ru

Яндекс Карты

flapenguin.me/talks/wasm-uncut

github.com/flapenguin/holyjs-wasm-uncut-examples

Яндекс Карты

flapenguin.me/talks/wasm-uncut

github.com/flapenguin/holyjs-wasm-uncut-examples

Спасибо

Роенко Андрей

Руководитель группы разработки JavaScript API Яндекс.Карт

flapenguin@yandex.ru

@flapenguin

flapenguin

flapenguin.me