Class Module

A WASM module.

`Module` itself is:

- an instantiable class (via `new Module()`)
- a namespace containing the following members, which themselves are classes (see related documentation):
  - Module.Tag
  - Module.Global
  - Module.Memory
  - Module.Table
  - Module.Function
  - Module.DataSegment
  - Module.ElementSegment
  - Module.Import
  - Module.Export

Each instance of `Module`:

- is a WASM module with module manipulation methods (`.emitText()`, `.validate()`, etc.).
- is an individual namespace containing mixins, each with its own component manipulation methods (`.tags.add()`, `.globals.get()`, etc):
  - Module#tags
  - Module#globals
  - Module#memories
  - Module#tables
  - Module#functions
  - Module#dataSegments
  - Module#elementSegments
  - Module#imports
  - Module#exports
- has a property `.wasm`, a namespace for creating expressions in the module (`.wasm.nop()`, `.wasm.i32.add()`, etc.)

Defined in classes/module/Module.ts:151
Defined in classes/module/Module.ts:523

Index

Debugging

- (M) addDebugInfoFileName
- (M) getDebugInfoFileName
- (M) setDebugLocation

Emission & Execution

- (M) dispose
- (M) emitAsmjs
- (M) emitBinary
- (M) emitStackIR
- (M) emitText
- (M) interpret

Expression Manipulation

- (P) wasm
- (M) copyExpression
- (M) getSideEffects
- (M) pop

Module Component Operations

- (P) dataSegments
- (P) elementSegments
- (P) exports
- (P) functions
- (P) globals
- (P) imports
- (P) memories
- (P) tables
- (P) tags
- (A) features
- (A) start
- (M) getDataSegmentInfo

Ⓜ getMemoryInfo

Other

Ⓒ constructor
Ⓜ ~~addActiveElementSegment~~
Ⓜ addCustomSection
Ⓜ ~~addFunction~~
Ⓜ ~~addFunctionExport~~
Ⓜ ~~addFunctionImport~~
Ⓜ ~~addGlobal~~
Ⓜ ~~addGlobalExport~~
Ⓜ ~~addGlobalImport~~
Ⓜ ~~addMemoryExport~~
Ⓜ ~~addMemoryImport~~
Ⓜ ~~addPassiveElementSegment~~
Ⓜ ~~addTable~~
Ⓜ ~~addTableExport~~
Ⓜ ~~addTableImport~~
Ⓜ ~~addTag~~
Ⓜ ~~addTagExport~~
Ⓜ ~~addTagImport~~
Ⓜ ~~getDataSegment~~
Ⓜ ~~getDataSegmentByIndex~~
Ⓜ ~~getElementSegment~~
Ⓜ ~~getElementSegmentByIndex~~
Ⓜ ~~getExport~~
Ⓜ ~~getExportByIndex~~
Ⓜ ~~getFeatures~~
Ⓜ ~~getFunction~~
Ⓜ ~~getFunctionByIndex~~
Ⓜ ~~getGlobal~~
Ⓜ ~~getGlobalByIndex~~
Ⓜ ~~getNumDataSegments~~
Ⓜ ~~getNumElementSegments~~

- ~~getNumExports~~
- ~~getNumFunctions~~
- ~~getNumGlobals~~
- ~~getNumTables~~
- ~~getStart~~
- ~~getTable~~
- ~~getTableByIndex~~
- ~~getTableSegments~~
- ~~getTag~~
- ~~hasMemory~~
- ~~removeElementSegment~~
- ~~removeExport~~
- ~~removeFunction~~
- ~~removeGlobal~~
- ~~removeTable~~
- ~~removeTag~~
- ~~setFeatures~~
- setFieldName
- ~~setMemory~~
- ~~setStart~~
- setTypeName
- updateMaps

Validation & Optimization

- optimize
- optimizeFunction
- runPasses
- runPassesOnFunction
- validate

Debugging

addDebugInfoFileName

```
addDebugInfoFileName(filename: string): number
```

Adds a debug info file name to the module and returns its index.

Parameters

- filename: *string*

Returns *number*

Defined in classes/module/Module.ts:470

getDebugInfoFileName

```
getDebugInfoFileName(index: number): string
```

Gets the name of the debug info file at the specified index.

Parameters

- index: *number*

Returns *string*

Defined in classes/module/Module.ts:478

setDebugLocation

```
setDebugLocation(
    func: number,
    expr: number,
    fileIndex: number,
    lineNumber: number,
    columnNumber: number,
): void
```

Sets the debug location of the specified `ExpressionRef` within the specified `FunctionRef`.

Parameters

- func: *number*
- expr: *number*
- fileIndex: *number*
- lineNumber: *number*
- columnNumber: *number*

Returns *void*

Defined in classes/module/Module.ts:486

Emission & Execution

dispose

```
dispose(): void
```

Releases the resources held by the module once it isn't needed anymore.

Returns *void*

Defined in classes/module/Module.ts:414

emitAsmjs

```
emitAsmjs(): string
```

Returns the asm.js representation of the module.

Returns *string*

Defined in classes/module/Module.ts:359

emitBinary

```
emitBinary(): Uint8Array
```

Returns the module in binary format.

Returns *Uint8Array*

Defined in classes/module/Module.ts:374

```
emitBinary(sourceMapUrl: string): { binary: Uint8Array; sourceMap: string }
```

Returns the module in binary format with a given source map.

Parameters

- sourceMapUrl: *string*

Returns { **binary**: *Uint8Array*; **sourceMap**: *string* }

Defined in classes/module/Module.ts:379

emitStackIR

```
emitStackIR(): string
```

Returns the module in official stack-style text format.

Returns *string*

Defined in classes/module/Module.ts:346

emitText

```
emitText(): string
```

Returns the module in Binaryen's s-expression text format (not official stack-style text format).

Returns *string*

Defined in classes/module/Module.ts:333

interpret

```
interpret(): void
```

Runs the module in the interpreter, calling the start function.

Returns *void*

Defined in classes/module/Module.ts:406

Expression Manipulation

`Readonly` wasm

This module's WASM expression builder.

See ExpressionBuilder for its type signature.

N.B.: For convenience, developers may want to destructure the module to free `wasm`:

```
const mod = new Module();
const {wasm} = mod;
wasm.drop(wasm.i32.add(wasm.i32.const(3), wasm.i32.const(5)));
```

or to free its properties:

```
const {i32, drop} = mod.wasm;
drop(i32.add(i32.const(3), i32.const(5)));
```

Defined in classes/module/Module.ts:188

copyExpression

```
copyExpression(expr: number): number
```

Creates a deep copy of an expression.

Parameters

- expr: *number*

Returns *number*

Defined in classes/module/Module.ts:228

getSideEffects

```
getSideEffects(expr: number): SideEffect
```

Gets the side effects of the specified expression.

Parameters

- expr: *number*

Returns *SideEffect*

Defined in classes/module/Module.ts:220

pop

```
pop(typ: number): number
```

Pseudo-instruction enabling Binaryen to reason about multiple values on the stack.

Parameters

- typ: *number*

Returns *number*

Defined in classes/module/Module.ts:194

Module Component Operations

`Readonly` dataSegments

```
dataSegments: {
    count(): number;
    get(name: string): number;
    getByIndex(index: number): number;
} = ...
```

Type Declaration

- **count**: **function**

  ```
  count(): number
  ```

  Gets the number of data segments within the module.

  Returns *number*

  Defined in classes/module/DataSegment.ts:70

- **get**: **function**

  ```
  get(name: string): number
  ```

  Gets a data segment by name.

  Parameters

  - name: *string*

  Returns *number*

  Defined in classes/module/DataSegment.ts:60

- **getByIndex**: **function**

  ```
  getByIndex(index: number): number
  ```

  Gets a data segment by index.

  Parameters

  - index: *number*

  Returns *number*

Defined in classes/module/DataSegment.ts:65

Defined in classes/module/Module.ts:239

`Readonly` elementSegments

```
elementSegments: {
    addActive(
        table: string,
        name: string,
        funcNames: readonly string[],
        offset?: number,
    ): number;
    addPassive(name: string, funcNames: readonly string[]): number;
    count(): number;
    get(name: string): number;
    getByIndex(index: number): number;
    remove(name: string): void;
} = ...
```

Type Declaration

- **addActive**: **function**

  ```
  addActive(
      table: string,
      name: string,
      funcNames: readonly string[],
      offset?: number,
  ): number
  ```

  Adds an active element segment.

  #### Parameters

  - table: *string*
  - name: *string*
  - funcNames: readonly *string*[]
  - offset: *number* = ...

  #### Returns *number*

  Defined in classes/module/ElementSegment.ts:56

- **addPassive**: **function**

```
addPassive(name: string, funcNames: readonly string[]): number
```

Adds a passive element segment.

Parameters

- name: *string*
- funcNames: readonly *string*[]

Returns *number*

Defined in classes/module/ElementSegment.ts:68

- **count**: **function**

```
count(): number
```

Gets the number of element segments within the module.

Returns *number*

Defined in classes/module/ElementSegment.ts:88

- **get**: **function**

```
get(name: string): number
```

Gets an element segment by name.

Parameters

- name: *string*

Returns *number*

Defined in classes/module/ElementSegment.ts:78

- **getByIndex**: **function**

```
getByIndex(index: number): number
```

Gets an element segment by index.

Parameters

- index: *number*

Returns *number*

Defined in classes/module/ElementSegment.ts:83

- **remove**: **function**

```
remove(name: string): void
```

Removes an element segment by name.

Parameters

- name: *string*

Returns *void*

Defined in classes/module/ElementSegment.ts:93

Defined in classes/module/Module.ts:240

`Readonly` exports

```
exports: {
    addFunction(internalName: string, externalName: string): number;
    addGlobal(internalName: string, externalName: string): number;
    addMemory(internalName: string, externalName: string): number;
    addTable(internalName: string, externalName: string): number;
    addTag(internalName: string, externalName: string): number;
    count(): number;
    get(externalName: string): number;
    getByIndex(index: number): number;
    remove(externalName: string): void;
} = ...
```

Type Declaration

- **addFunction**: **function**

```
addFunction(internalName: string, externalName: string): number
```

Adds a function export.

Parameters

- internalName: *string*
- externalName: *string*

Returns *number*

Defined in classes/module/Export.ts:86

- **addGlobal**: **function**

```
addGlobal(internalName: string, externalName: string): number
```

Adds a global variable export. Exported globals must be immutable.

Parameters

- internalName: *string*
- externalName: *string*

Returns *number*

Defined in classes/module/Export.ts:71

- **addMemory**: **function**

```
addMemory(internalName: string, externalName: string): number
```

Adds a memory export. There's just one memory for now, using name "0".

Parameters

- internalName: *string*
- externalName: *string*

Returns *number*

Defined in classes/module/Export.ts:76

- **addTable**: **function**

```
addTable(internalName: string, externalName: string): number
```

Adds a table export. There's just one table for now, using name "0".

Parameters

- internalName: *string*
- externalName: *string*

Returns *number*

Defined in classes/module/Export.ts:81

- **addTag**: **function**

```
addTag(internalName: string, externalName: string): number
```

Adds a tag export.

Parameters

- internalName: *string*
- externalName: *string*

Returns *number*

Defined in classes/module/Export.ts:66

- **count**: **function**

```
count(): number
```

Gets the number of exports witin the module.

Returns *number*

Defined in classes/module/Export.ts:56

- **get**: **function**

```
get(externalName: string): number
```

Gets an export by name.

Parameters

- externalName: *string*

Returns *number*

Defined in classes/module/Export.ts:46

- **getByIndex**: **function**

```
getByIndex(index: number): number
```

Gets an export by index.

Parameters

- index: *number*

Returns *number*

Defined in classes/module/Export.ts:51

- **remove**: **function**

```
remove(externalName: string): void
```

Removes an export, by external name.

Parameters

- externalName: *string*

Returns *void*

Defined in classes/module/Export.ts:61

Defined in classes/module/Module.ts:242

`Readonly` functions

```
functions: {
    add(
        name: string,
        params: number,
        results: number,
        varTypes: readonly number[],
        body: number,
    ): number;
    count(): number;
    get(name: string): number;
    getByIndex(index: number): number;
    remove(name: string): void;
} = ...
```

Type Declaration

- **add**: **function**

  ```
  add(
      name: string,
      params: number,
      results: number,
      varTypes: readonly number[],
      body: number,
  ): number
  ```

  Adds a function. `varTypes` indicate additional locals, in the given order.

  #### Parameters

  - name: *string*
  - params: *number*
  - results: *number*
  - varTypes: readonly *number*[]
  - body: *number*

  #### Returns *number*

  Defined in classes/module/Function.ts:98

- **count**: **function**

  ```
  count(): number
  ```

  Gets the number of functions within the module.

  #### Returns *number*

Defined in classes/module/Function.ts:121

- **get**: **function**

```
get(name: string): number
```

Gets a function by name.

Parameters

- name: *string*

Returns *number*

Defined in classes/module/Function.ts:111

- **getByIndex**: **function**

```
getByIndex(index: number): number
```

Gets a function by index.

Parameters

- index: *number*

Returns *number*

Defined in classes/module/Function.ts:116

- **remove**: **function**

```
remove(name: string): void
```

Removes a function by name.

Parameters

- name: *string*

Returns *void*

Defined in classes/module/Function.ts:126

Defined in classes/module/Module.ts:238

`Readonly` globals

```
globals: {
    add(name: string, type: number, mutable: boolean, init: number): number;
    count(): number;
    get(name: string): number;
    getByIndex(index: number): number;
    remove(name: string): void;
} = ...
```

Type Declaration

- **add**: **function**

  ```
  add(name: string, type: number, mutable: boolean, init: number): number
  ```

  Adds a global instance variable.

  #### Parameters

  - name: *string*
  - type: *number*
  - mutable: *boolean*
  - init: *number*

  Returns *number*

  Defined in classes/module/Global.ts:53

- **count**: **function**

  ```
  count(): number
  ```

  Gets the number of globals within the module.

  Returns *number*

  Defined in classes/module/Global.ts:68

- **get**: **function**

```
get(name: string): number
```

Gets a global by name.

Parameters

- name: *string*

Returns *number*

Defined in classes/module/Global.ts:58

- **getByIndex**: **function**

```
getByIndex(index: number): number
```

Gets a global by index.

Parameters

- index: *number*

Returns *number*

Defined in classes/module/Global.ts:63

- **remove**: **function**

```
remove(name: string): void
```

Removes a global by name.

Parameters

- name: *string*

Returns *void*

Defined in classes/module/Global.ts:73

Defined in classes/module/Module.ts:235

`Readonly` imports

```
imports: {
    addFunction(
        internalName: string,
        externalModuleName: string,
        externalBaseName: string,
        params: number,
        results: number,
    ): void;
    addGlobal(
        internalName: string,
        externalModuleName: string,
        externalBaseName: string,
        globalType: number,
        mutable: boolean,
    ): void;
    addMemory(
        internalName: string,
        externalModuleName: string,
        externalBaseName: string,
        shared: boolean,
    ): void;
    addTable(
        internalName: string,
        externalModuleName: string,
        externalBaseName: string,
    ): void;
    addTag(
        internalName: string,
        externalModuleName: string,
        externalBaseName: string,
        params: number,
        results: number,
    ): void;
} = ...
```

Type Declaration

- **addFunction**: **function**

  ```
  addFunction(
      internalName: string,
      externalModuleName: string,
      externalBaseName: string,
      params: number,
      results: number,
  ): void
  ```

  Adds a function import.

  #### Parameters

- internalName: *string*
- externalModuleName: *string*
- externalBaseName: *string*
- params: *number*
- results: *number*

Returns *void*

Defined in classes/module/Import.ts:55

- **addGlobal**: **function**

```
addGlobal(
    internalName: string,
    externalModuleName: string,
    externalBaseName: string,
    globalType: number,
    mutable: boolean,
): void
```

Adds a global variable import. Imported globals must be immutable.

Parameters

- internalName: *string*
- externalModuleName: *string*
- externalBaseName: *string*
- globalType: *number*
- mutable: *boolean*

Returns *void*

Defined in classes/module/Import.ts:40

- **addMemory**: **function**

```
addMemory(
    internalName: string,
    externalModuleName: string,
    externalBaseName: string,
    shared: boolean,
): void
```

Adds a memory import. There's just one memory for now, using name "0".

Parameters

- internalName: *string*

- externalModuleName: *string*
- externalBaseName: *string*
- shared: *boolean*

Returns *void*

Defined in classes/module/Import.ts:45

- **addTable**: **function**

```
addTable(
    internalName: string,
    externalModuleName: string,
    externalBaseName: string,
): void
```

Adds a table import. There's just one table for now, using name "0".

Parameters

- internalName: *string*
- externalModuleName: *string*
- externalBaseName: *string*

Returns *void*

Defined in classes/module/Import.ts:50

- **addTag**: **function**

```
addTag(
    internalName: string,
    externalModuleName: string,
    externalBaseName: string,
    params: number,
    results: number,
): void
```

Adds a tag import.

Parameters

- internalName: *string*
- externalModuleName: *string*
- externalBaseName: *string*
- params: *number*
- results: *number*

Returns *void*

Defined in classes/module/Import.ts:35

Defined in classes/module/Module.ts:241

`Readonly` memories

```
memories: {
    has(): boolean;
    set(
        initial: number,
        maximum: number,
        exportName: string,
        segments?: readonly {
            data: Uint8Array;
            name?: string;
            offset: number;
            passive: boolean;
        }[],
        shared?: boolean,
        memory64?: boolean,
        internalName?: string,
    ): void;
} = ...
```

Type Declaration

- **has**: **function**

  ```
  has(): boolean
  ```

  Returns whether the module has a memory.

  ### Returns *boolean*

  Defined in classes/module/Memory.ts:116

- **set**: **function**

  ```
  set(
      initial: number,
      maximum: number,
      exportName: string,
      segments?: readonly {
          data: Uint8Array;
          name?: string;
  ```

```
        offset: number;
        passive: boolean;
    }[],
    shared?: boolean,
    memory64?: boolean,
    internalName?: string,
): void
```

Sets the memory. There's just one memory for now, using name "0". Providing `exportName` also creates a memory export.

Parameters

- initial: *number*
- maximum: *number*
- exportName: *string*
- segments: readonly { data: *Uint8Array*; name?: *string*; offset: *number*; passive: *boolean* }[] = []
- shared: *boolean* = false
- memory64: *boolean* = false
- `Optional` internalName: *string*

Returns *void*

Defined in classes/module/Memory.ts:70

Defined in classes/module/Module.ts:236

`Readonly` tables

```
tables: {
    add(
        name: string,
        initial: number,
        maximum: number,
        type?: number,
        init?: number,
    ): number;
    count(): number;
    get(name: string): number;
    getByIndex(index: number): number;
    getSegments(table: number): number[];
    remove(name: string): void;
} = ...
```

Type Declaration

- **add**: **function**

```
add(
    name: string,
    initial: number,
    maximum: number,
    type?: number,
    init?: number,
): number
```

Adds a table.

Parameters

- name: *string*
- initial: *number*
- maximum: *number*
- type: *number* = funcref
- `Optional` init: *number*

Returns *number*

Defined in classes/module/Table.ts:79

- **count**: **function**

```
count(): number
```

Gets the number of tables within the module.

Returns *number*

Defined in classes/module/Table.ts:109

- **get**: **function**

```
get(name: string): number
```

Gets a table by name.

Parameters

- name: *string*

Returns *number*

Defined in classes/module/Table.ts:84

- **getByIndex**: **function**

  ```
  getByIndex(index: number): number
  ```

  Gets a table by index.

  Parameters

  - index: *number*

  Returns *number*

  Defined in classes/module/Table.ts:89

- **getSegments**: **function**

  ```
  getSegments(table: number): number[]
  ```

  Gets the number of table segments within the module.

  Parameters

  - table: *number*

  Returns *number*[]

  Defined in classes/module/Table.ts:94

- **remove**: **function**

  ```
  remove(name: string): void
  ```

  Removes a table by name.

  Parameters

  - name: *string*

  Returns *void*

  Defined in classes/module/Table.ts:114

Defined in classes/module/Module.ts:237

`Readonly` tags

```
tags: {
    add(name: string, params: number, results: number): number;
    get(name: string): number;
    remove(name: string): void;
} = ...
```

Type Declaration

- **add**: **function**

  ```
  add(name: string, params: number, results: number): number
  ```

  Adds a tag.

  #### Parameters

  - name: *string*
  - params: *number*
  - results: *number*

  Returns *number*

  Defined in classes/module/Tag.ts:50

- **get**: **function**

  ```
  get(name: string): number
  ```

  Gets a tag by name.

  #### Parameters

  - name: *string*

  Returns *number*

  Defined in classes/module/Tag.ts:55

- **remove**: **function**

```
remove(name: string): void
```

Removes a tag by name.

Parameters

- name: *string*

Returns *void*

Defined in classes/module/Tag.ts:60

Defined in classes/module/Module.ts:234

features

```
get features(): Feature
```

The WebAssembly features enabled for this module. Features are a bitmask of `Feature` enum members.

Returns *Feature*

Defined in classes/module/Module.ts:261

```
set features(features: Feature): void
```

Parameters

- features: *Feature*

Returns *void*

Defined in classes/module/Module.ts:262

start

```
get start(): number
```

The start function.

Returns *number*

Defined in classes/module/Module.ts:253

```
set start(start: number): void
```

Parameters

- start: *number*

Returns *void*

Defined in classes/module/Module.ts:254

getDataSegmentInfo

```
getDataSegmentInfo(segment: number): DataSegment
```

Parameters

- segment: *number*

Returns *DataSegment*

Defined in classes/module/Module.ts:323

getMemoryInfo

```
getMemoryInfo(name: string): Memory
```

Parameters

- name: *string*

Returns *Memory*

Defined in classes/module/Module.ts:318

Other

constructor

```
new Module(): Module
```

Returns *Module*

~~addActiveElementSegment~~

```
addActiveElementSegment(
    table: string,
    name: string,
    funcNames: readonly string[],
    offset: number,
): number
```

Parameters

- table: *string*
- name: *string*
- funcNames: readonly *string*[]
- offset: *number*

Returns *number*

Deprecated

Use `this.elementSegments.addActive` instead.

Defined in classes/module/Module.ts:294

addCustomSection

```
addCustomSection(name: string, contents: Uint8Array): void
```

Adds a custom section to the binary.

Parameters

- name: *string*
- contents: *Uint8Array*

Returns *void*

Defined in classes/module/Module.ts:502

~~addFunction~~

```
addFunction(
    name: string,
    params: number,
    results: number,
    varTypes: readonly number[],
    body: number,
): number
```

Parameters

- name: *string*
- params: *number*
- results: *number*
- varTypes: readonly *number*[]
- body: *number*

Returns *number*

Deprecated

Use `this.functions.add` instead.

Defined in classes/module/Module.ts:284

~~addFunctionExport~~

```
addFunctionExport(internalName: string, externalName: string): number
```

Parameters

- internalName: *string*
- externalName: *string*

Returns *number*

Deprecated

Use `this.exports.addFunction` instead.

Defined in classes/module/Module.ts:315

~~addFunctionImport~~

```
addFunctionImport(
    internalName: string,
    externalModuleName: string,
    externalBaseName: string,
    params: number,
    results: number,
): void
```

Parameters

- internalName: *string*
- externalModuleName: *string*
- externalBaseName: *string*
- params: *number*
- results: *number*

Returns *void*

Deprecated

Use `this.imports.addFunction` instead.

Defined in classes/module/Module.ts:305

~~addGlobal~~

```
addGlobal(name: string, type: number, mutable: boolean, init: number): number
```

Parameters

- name: *string*

- type: *number*
- mutable: *boolean*
- init: *number*

Returns *number*

Deprecated

Use `this.globals.add` instead.

Defined in classes/module/Module.ts:268

~~addGlobalExport~~

```
addGlobalExport(internalName: string, externalName: string): number
```

Parameters

- internalName: *string*
- externalName: *string*

Returns *number*

Deprecated

Use `this.exports.addGlobal` instead.

Defined in classes/module/Module.ts:312

~~addGlobalImport~~

```
addGlobalImport(
    internalName: string,
    externalModuleName: string,
    externalBaseName: string,
    globalType: number,
    mutable: boolean,
): void
```

Parameters

- internalName: *string*

- externalModuleName: *string*
- externalBaseName: *string*
- globalType: *number*
- mutable: *boolean*

Returns *void*

Deprecated

Use `this.imports.addGlobal` instead.

Defined in classes/module/Module.ts:302

~~addMemoryExport~~

```
addMemoryExport(internalName: string, externalName: string): number
```

Parameters

- internalName: *string*
- externalName: *string*

Returns *number*

Deprecated

Use `this.exports.addMemory` instead.

Defined in classes/module/Module.ts:313

~~addMemoryImport~~

```
addMemoryImport(
    internalName: string,
    externalModuleName: string,
    externalBaseName: string,
    shared: boolean,
): void
```

Parameters

- internalName: *string*

- externalModuleName: *string*
- externalBaseName: *string*
- shared: *boolean*

Returns *void*

Deprecated

Use `this.imports.addMemory` instead.

Defined in classes/module/Module.ts:303

~~addPassiveElementSegment~~

```
addPassiveElementSegment(name: string, funcNames: readonly string[]): number
```

Parameters

- name: *string*
- funcNames: readonly *string*[]

Returns *number*

Deprecated

Use `this.elementSegments.addPassive` instead.

Defined in classes/module/Module.ts:295

~~addTable~~

```
addTable(
    name: string,
    initial: number,
    maximum: number,
    type?: number,
    init?: number,
): number
```

Parameters

- name: *string*
- initial: *number*
- maximum: *number*
- type: *number* = funcref
- `Optional` init: *number*

Returns *number*

Deprecated

Use `this.tables.add` instead.

Defined in classes/module/Module.ts:277

~~addTableExport~~

```
addTableExport(internalName: string, externalName: string): number
```

Parameters

- internalName: *string*
- externalName: *string*

Returns *number*

Deprecated

Use `this.exports.addTable` instead.

Defined in classes/module/Module.ts:314

~~addTableImport~~

```
addTableImport(
    internalName: string,
    externalModuleName: string,
    externalBaseName: string,
): void
```

Parameters

- internalName: *string*

- externalModuleName: *string*
- externalBaseName: *string*

Returns *void*

Deprecated

Use `this.imports.addTable` instead.

Defined in classes/module/Module.ts:304

~~addTag~~

```
addTag(name: string, params: number, results: number): number
```

Parameters

- name: *string*
- params: *number*
- results: *number*

Returns *number*

Deprecated

Use `this.tags.add` instead.

Defined in classes/module/Module.ts:264

~~addTagExport~~

```
addTagExport(internalName: string, externalName: string): number
```

Parameters

- internalName: *string*
- externalName: *string*

Returns *number*

Deprecated

Use `this.exports.addTag` instead.

Defined in classes/module/Module.ts:311

~~addTagImport~~

```
addTagImport(
    internalName: string,
    externalModuleName: string,
    externalBaseName: string,
    params: number,
    results: number,
): void
```

Parameters

- internalName: *string*
- externalModuleName: *string*
- externalBaseName: *string*
- params: *number*
- results: *number*

Returns *void*

Deprecated

Use `this.imports.addTag` instead.

Defined in classes/module/Module.ts:301

~~getDataSegment~~

```
getDataSegment(name: string): number
```

Parameters

- name: *string*

Returns *number*

Deprecated

Use `this.dataSegments.get` instead.

Defined in classes/module/Module.ts:290

~~getDataSegmentByIndex~~

```
getDataSegmentByIndex(index: number): number
```

Parameters

- index: *number*

Returns *number*

Deprecated

Use `this.dataSegments.getByIndex` instead.

Defined in classes/module/Module.ts:291

~~getElementSegment~~

```
getElementSegment(name: string): number
```

Parameters

- name: *string*

Returns *number*

Deprecated

Use `this.elementSegments.get` instead.

Defined in classes/module/Module.ts:296

~~getElementSegmentByIndex~~

```
getElementSegmentByIndex(index: number): number
```

Parameters

- index: *number*

Returns *number*

Deprecated

Use `this.elementSegments.getByIndex` instead.

Defined in classes/module/Module.ts:297

~~getExport~~

```
getExport(externalName: string): number
```

Parameters

- externalName: *string*

Returns *number*

Deprecated

Use `this.exports.get` instead.

Defined in classes/module/Module.ts:307

~~getExportByIndex~~

```
getExportByIndex(index: number): number
```

Parameters

- index: *number*

Returns *number*

Deprecated

Use `this.exports.getByIndex` instead.

Defined in classes/module/Module.ts:308

~~getFeatures~~

```
getFeatures(): Feature
```

Returns *Feature*

Deprecated

Use `this.features` instead.

Defined in classes/module/Module.ts:246

~~getFunction~~

```
getFunction(name: string): number
```

Parameters

- name: *string*

Returns *number*

Deprecated

Use `this.functions.get` instead.

Defined in classes/module/Module.ts:285

~~getFunctionByIndex~~

```
getFunctionByIndex(index: number): number
```

Parameters

- index: *number*

Returns *number*

Deprecated

Use `this.functions.getByIndex` instead.

Defined in classes/module/Module.ts:286

~~getGlobal~~

```
getGlobal(name: string): number
```

Parameters

- name: *string*

Returns *number*

Deprecated

Use `this.globals.get` instead.

Defined in classes/module/Module.ts:269

~~getGlobalByIndex~~

```
getGlobalByIndex(index: number): number
```

Parameters

- index: *number*

Returns *number*

Deprecated

Use `this.globals.getByIndex` instead.

Defined in classes/module/Module.ts:270

~~getNumDataSegments~~

```
getNumDataSegments(): number
```

Returns *number*

Deprecated

Use `this.dataSegments.count` instead.

Defined in classes/module/Module.ts:292

~~getNumElementSegments~~

```
getNumElementSegments(): number
```

Returns *number*

Deprecated

Use `this.elementSegments.count` instead.

Defined in classes/module/Module.ts:298

~~getNumExports~~

```
getNumExports(): number
```

Returns *number*

Deprecated

Use `this.exports.count` instead.

Defined in classes/module/Module.ts:309

~~getNumFunctions~~

```
getNumFunctions(): number
```

Returns *number*

Deprecated

Use `this.functions.count` instead.

Defined in classes/module/Module.ts:287

~~getNumGlobals~~

```
getNumGlobals(): number
```

Returns *number*

Deprecated

Use `this.globals.count` instead.

Defined in classes/module/Module.ts:271

~~getNumTables~~

```
getNumTables(): number
```

Returns *number*

Deprecated

Use `this.tables.count` instead.

Defined in classes/module/Module.ts:281

~~getStart~~

```
getStart(): number
```

Returns *number*

Deprecated

Use `this.start` instead.

Defined in classes/module/Module.ts:244

~~getTable~~

```
getTable(name: string): number
```

Parameters

- name: *string*

Returns *number*

Deprecated

Use `this.tables.get` instead.

Defined in classes/module/Module.ts:278

~~getTableByIndex~~

```
getTableByIndex(index: number): number
```

Parameters

- index: *number*

Returns *number*

Deprecated

Use `this.tables.getByIndex` instead.

Defined in classes/module/Module.ts:279

~~getTableSegments~~

```
getTableSegments(table: number): number[]
```

Parameters

- table: *number*

Returns *number*[]

Deprecated

Use `this.tables.getSegments` instead.

Defined in classes/module/Module.ts:280

~~getTag~~

```
getTag(name: string): number
```

Parameters

- name: *string*

Returns *number*

Deprecated

Use `this.tags.get` instead.

Defined in classes/module/Module.ts:265

~~hasMemory~~

```
hasMemory(): boolean
```

Returns *boolean*

Deprecated

Use `this.memories.has` instead.

Defined in classes/module/Module.ts:275

~~removeElementSegment~~

```
removeElementSegment(name: string): void
```

Parameters

- name: *string*

Returns *void*

Deprecated

Use `this.elementSegments.remove` instead.

Defined in classes/module/Module.ts:299

~~removeExport~~

```
removeExport(externalName: string): void
```

Parameters

- externalName: *string*

Returns *void*

Deprecated

Use `this.exports.remove` instead.

Defined in classes/module/Module.ts:310

~~removeFunction~~

```
removeFunction(name: string): void
```

Parameters

- name: *string*

Returns *void*

Deprecated

Use `this.functions.remove` instead.

Defined in classes/module/Module.ts:288

~~removeGlobal~~

```
removeGlobal(name: string): void
```

Parameters

- name: *string*

Returns *void*

Deprecated

Use `this.globals.remove` instead.

Defined in classes/module/Module.ts:272

~~removeTable~~

```
removeTable(name: string): void
```

Parameters

- name: *string*

Returns *void*

Deprecated

Use `this.tables.remove` instead.

Defined in classes/module/Module.ts:282

~~removeTag~~

```
removeTag(name: string): void
```

Parameters

- name: *string*

Returns *void*

Deprecated

Use `this.tags.remove` instead.

Defined in classes/module/Module.ts:266

~~setFeatures~~

```
setFeatures(features: Feature): Feature
```

Parameters

- features: *Feature*

Returns *Feature*

Deprecated

Use `this.features` instead.

Defined in classes/module/Module.ts:247

setFieldName

```
setFieldName(heapType: number, index: number, name: string): void
```

[description]

Parameters

- heapType: *number*
- index: *number*
- name: *string*

Returns *void*

Defined in classes/module/Module.ts:497

~~setMemory~~

```
setMemory(
    initial: number,
    maximum: number,
    exportName: string,
    segments?: readonly any[],
    shared?: boolean,
    memory64?: boolean,
    internalName?: string,
): void
```

Parameters

- initial: *number*
- maximum: *number*
- exportName: *string*
- `Optional` segments: readonly *any*[]
- `Optional` shared: *boolean*

- `Optional` memory64: *boolean*
- `Optional` internalName: *string*

Returns *void*

Deprecated

Use `this.memories.set` instead.

Defined in classes/module/Module.ts:274

~~setStart~~

```
setStart(start: number): void
```

Parameters

- start: *number*

Returns *void*

Deprecated

Use `this.start` instead.

Defined in classes/module/Module.ts:245

setTypeName

```
setTypeName(heapType: number, name: string): void
```

[description]

Parameters

- heapType: *number*
- name: *string*

Returns *void*

Defined in classes/module/Module.ts:492

updateMaps

```
updateMaps(): void
```

Updates the internal name mapping logic in a module. This must be called after renaming module elements.

Returns *void*

Defined in classes/module/Module.ts:510

Validation & Optimization

optimize

```
optimize(): void
```

Optimizes the module using the default optimization passes.

Returns *void*

Defined in classes/module/Module.ts:431

optimizeFunction

```
optimizeFunction(func: string | number): void
```

Optimizes a single function using the default optimization passes.

Parameters

- func: *string* | *number*

Returns *void*

Defined in classes/module/Module.ts:439

runPasses

```
runPasses(passes: readonly string[]): void
```

Runs the specified passes on the module.

Parameters

- passes: readonly *string*[]

Returns *void*

Defined in classes/module/Module.ts:450

runPassesOnFunction

```
runPassesOnFunction(func: string | number, passes: readonly string[]): void
```

Runs the specified passes on a single function.

Parameters

- func: *string* | *number*
- passes: readonly *string*[]

Returns *void*

Defined in classes/module/Module.ts:458

validate

```
validate(): number
```

Validates the module. Returns `true` if valid, otherwise prints validation errors and returns `false`.

Returns *number*

Defined in classes/module/Module.ts:423