export function getModuleInstanceState(node: Node): ModuleInstanceState {

// A module is uninstantiated if it contains only

// 1. interface declarations, type alias declarations

if (node.kind === SyntaxKind.InterfaceDeclaration || node.kind === SyntaxKind.TypeAliasDeclaration) {

return ModuleInstanceState.NonInstantiated;

}

// 2. const enum declarations

else if (isConstEnumDeclaration(node)) {

return ModuleInstanceState.ConstEnumOnly;

}

// 3. non-exported import declarations

else if ((node.kind === SyntaxKind.ImportDeclaration || node.kind === SyntaxKind.ImportEqualsDeclaration) && !(node.flags & NodeFlags.Export)) {

return ModuleInstanceState.NonInstantiated;

}

// 4. other uninstantiated module declarations.

else if (node.kind === SyntaxKind.ModuleBlock) {

let state = ModuleInstanceState.NonInstantiated;

forEachChild(node, n => {

switch (getModuleInstanceState(n)) {

case ModuleInstanceState.NonInstantiated:

// child is non-instantiated - continue searching

return false;

case ModuleInstanceState.ConstEnumOnly:

// child is const enum only - record state and continue searching

state = ModuleInstanceState.ConstEnumOnly;

return false;

case ModuleInstanceState.Instantiated:

// child is instantiated - record state and stop

state = ModuleInstanceState.Instantiated;

return true;

}

});

return state;

}

else if (node.kind === SyntaxKind.ModuleDeclaration) {

return getModuleInstanceState((<ModuleDeclaration>node).body);

}

else {

return ModuleInstanceState.Instantiated;

}

}