Uniwersytet Rzeszowski
Kolegium Nauk Przyrodniczych
Instytut Informatyki
mgr inż. W. G., mgr inż. M. M., mgr inż. J. S.

JSDoc

Dokumentowanie kodu

Mając przykładową klasę:

```
js > example.js > SampleClass
1   class SampleClass {
2       publicFunction(a, b, c) {
3           // Logic for a public function
4           return a + b + c;
5       }
6
7       protectedFunction(...params) {
8           // Logic for a protected function
9       }
10
11      privateFunction() {
12          // Logic for a private function
13      }
14
15      undefinedFunction() {
16          // Logic for a function with undefined access (by default, public)
17      }
18  }
```

Aby wygenerować część kodu JSDoc wystarczy nad daną klasą zacząć komentarz wpisując /** po czym gdy menu kontekstowe pokaże następującą opcję wcisnąć enter:

```
js > example.js > ...
1   /** */
2   cla abc /** */                                    JSDoc comment
3       publicFunction(a, b, c) {
```

```
js > example.js > ...
1  /**
2   * 
3   */
4  class SampleClass {
5      publicFunction(a, b, c) {
6          // Logic for a public function
7          return a + b + c;
8      }
9
10     protectedFunction(...params) {
11         // Logic for a protected function
12     }
13
14     privateFunction() {
15         // Logic for a private function
16     }
17
18     undefinedFunction() {
19         // Logic for a function with undefined access (by default, public)
20     }
21 }
```

Następnie opisujemy zwięźle zadanie danej klasy i metod w niej zawartych oraz używamy odpowiedniej adnotacji.

```
js > example.js > SampleClass
1  /**
2   * This class represents a sample "Class".
3   *
4   * @class
5   */
6  class SampleClass {
7      publicFunction(a, b, c) {
8          // Logic for a public function
9          return a + b + c;
10     }
11
12     protectedFunction(...params) {
13         // Logic for a protected function
14     }
15
16     privateFunction() {
17         // Logic for a private function
18     }
19
20     undefinedFunction() {
21         // Logic for a function with undefined access (by default, public)
22     }
23 }
```

Listę adnotacji znajdziesz na stronie https://jsdoc.app/. Najważniejsze z nich to:

@private

@protected

@public

@abstract

@class lub @constructor – opis konstruktora

@classdesc – opis klasy

Class i classdesc mają inne zadania. Patrz https://jsdoc.app/tags-classdesc

@constant

@property – opis dla pola obiektu

@deprecated – funkcja/klasa, która kiedyś była używana, ale obecnie jest przestarzała

@example - przykład użycia funkcji/klasy

@async

@function – opis funkcji

@param – parametr funkcji

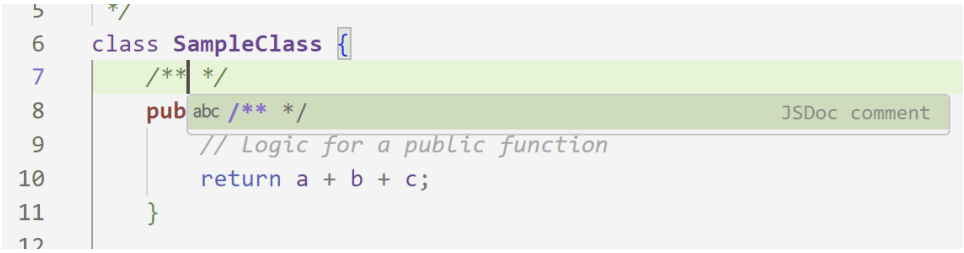
@returns – opis zwracanej wartości

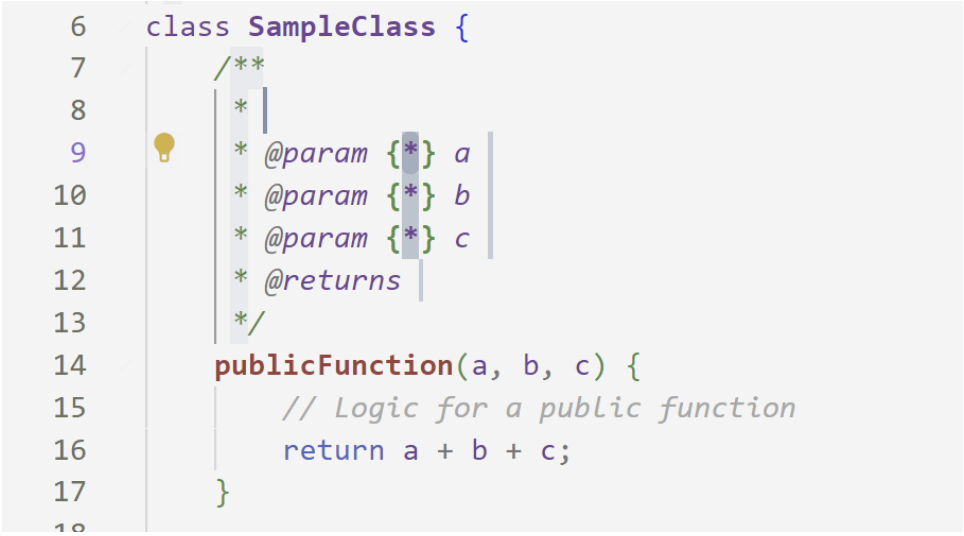
@throws – opis kiedy funkcja zwraca wyjątki i jakie

@todo – opis zadania do zrealizowania w kodzie

@ignore - pomiń w dokumentacji

VS Code potrafi odczytać niektóre z parametrów funkcji/obiektu i automatycznie wygenerować template dokumentacji. Przykładowo dla metody z poprzedniego przykładu:





Zostały rozpoznane parametry metody i ich nazwy.

Cechy dobrej dokumentacji

1. Klarowność i Czytelność:

Używaj jasnego, prostego języka. Unikaj zawiłych terminów, jeśli nie są konieczne. Dziel dokumentację na sekcje, takie jak opis, przykłady użycia, parametry, zwracane wartości itp.

2. Opis Funkcji / Klasy:

Rozpocznij od opisu ogólnego, wyjaśniającego przeznaczenie funkcji/klasy. Opisz, co funkcja robi lub jaki problem rozwiązuje.

3. Przykłady Użycia:

Dodaj realistyczne przykłady pokazujące, jak korzystać z funkcji/klasy. Pokaż różne przypadki użycia, aby użytkownik mógł łatwo zrozumieć, jak wykorzystać twój kod.

4. Parametry i Zwracane Wartości:

Dokładnie opisz oczekiwane parametry funkcji, ich typy i możliwe wartości. Wyjaśnij, co funkcja zwraca (wartości, obiekty, błędy) i jakie są ich znaczenia.

5. Tagi JSDoc lub Podobne:

Używaj odpowiednich tagów (np. @param, @returns) do opisywania parametrów i zwracanych wartości. Oznaczaj szczegóły dotyczące typów danych, opcji czy obiektów, aby ułatwić zrozumienie.

6. Aktualizacja i Utrzymanie:

Utrzymuj dokumentację na bieżąco, aktualizując ją, gdy kod ulega zmianom. Reaguj na opinie użytkowników i uzupełniaj dokumentację o rzeczy, które mogą być niejasne.

7. Przejrzyste Przykłady i Ilustracje:

Wykorzystuj grafiki, diagramy, tabele i inne formy wizualne, które mogą pomóc w zrozumieniu. Przykłady kodu powinny być czytelne, dobrze sformatowane i łatwe do skopiowania i wklejenia.

8. Dobrze Sformatowana Dokumentacja:

Używaj odpowiednich nagłówków, list, wcięć i formatowania tekstu, aby ułatwić czytanie. Starannie formatuj kod i fragmenty, aby były czytelne i łatwe do zrozumienia.

9. Testy:

Sprawdź dokumentację, wykonując testy, aby upewnić się, że opisy są zgodne z funkcjonalnością.

10. A przedewszystkim:

Projektuj dokumentację tak, aby była przyjazna dla nowych użytkowników, którzy mogą być niezaznajomieni z twoim kodem.

Konfiguracja narzędzi

1. Pobierz i zainstaluj środowisko NodeJS. Zalecana jest instalacja wersji LTS. Wersje rozwojowe zawierają najnowsze funkcje i ulepszenia, ale mogą być mniej stabilne, ponieważ nie otrzymują takiego samego stopnia testowania i stabilizacji co wersje LTS. Problemy w wersjach rozwojowych mogą występować i niekoniecznie są one natychmiast rozwiązywane.

   https://nodejs.org/en/download/

2. Po zakończeniu instalacji otwórz terminal (w systemie Windows może to być PowerShell lub wiersz poleceń) i wpisz poniższe komendy, aby sprawdzić zainstalowaną wersję Node.js. oraz wersję menedżera pakietów npm (Node Package Manager):

```
npm –v
node -v
```

```
C:\Users\wgalk>npm –v
9.8.1

C:\Users\wgalk>node –v
v18.13.0
```

3. Wykorzystując menedżer pakietów npm zaisntaluj narzędzie jsdoc wpisując w wierszu poleceń:

```
npm install -g jsdoc
```

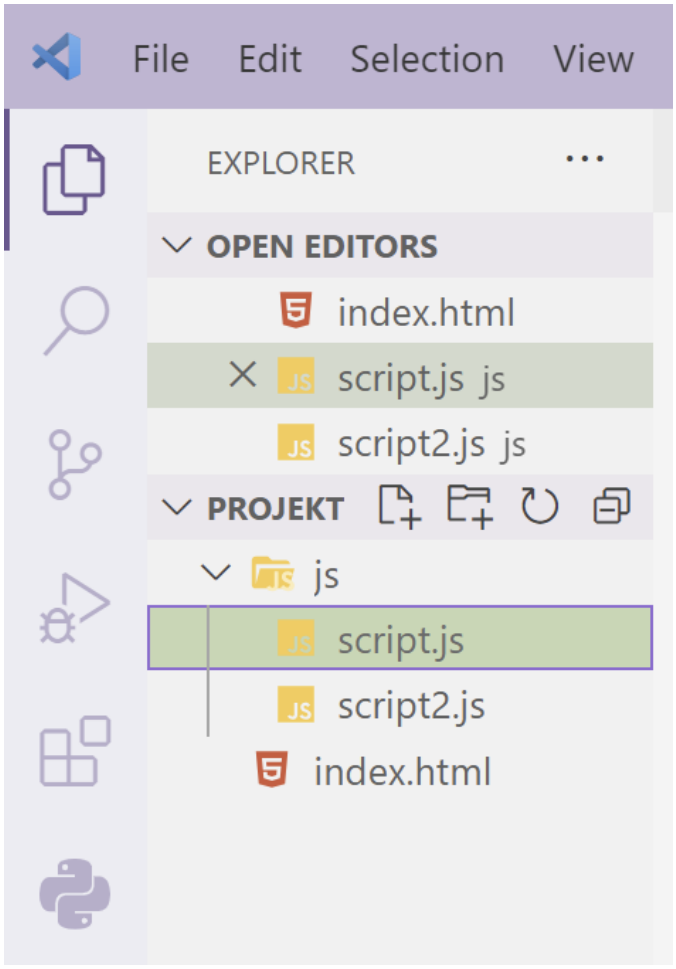
4. Sprawdź poprawność instalacji wpisując polecenie:

```
jsdoc -v
```

```
C:\Users\wgalk>jsdoc –v
JSDoc 4.0.2 (Sun, 19 Feb 2023 23:01:18 GMT)
```

Generowanie JSDOC

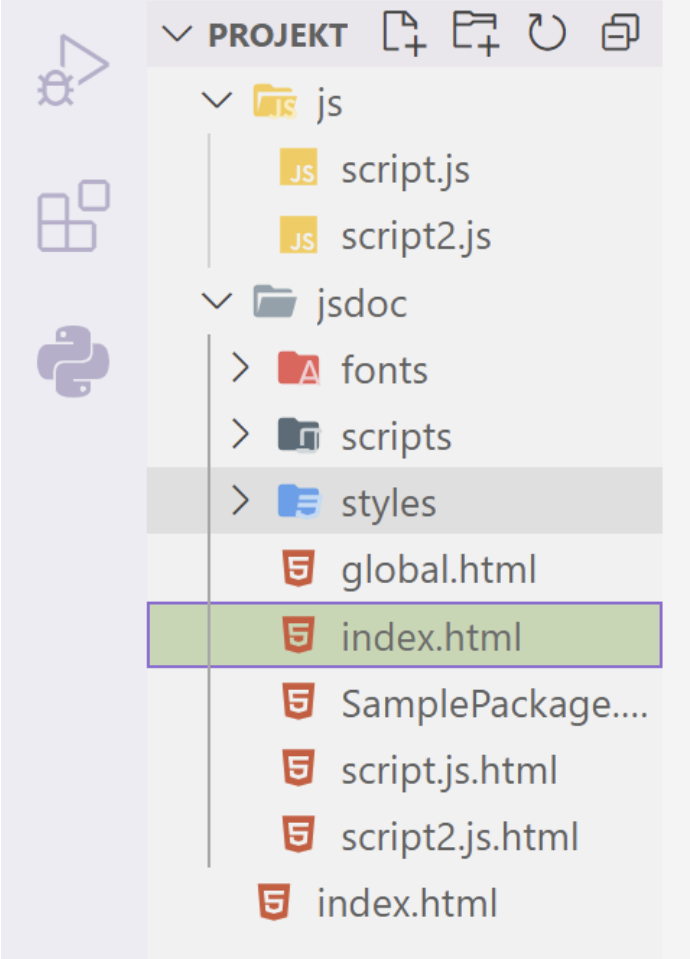
1. Wywołaj polecenie jsdoc wskazując folder ze skryptami JS, katalog w którym ma zostać utworzona strona dokumentacji, zakres dokumentacji (tworząc bibliotekę lub framework nie chcemy by użytkownicy zmieniali metody prywatne stąd są one domyślnie nieuwzględniane w dokumentacji. Na potrzeby projektu proszę zawrzeć również dokumentację metod prywatnych) Mając poniższe drzewo plików w projekcie chcąc wygenerować dokumentację dla pliku script.js polecenie będzie wyglądać jak poniżej:



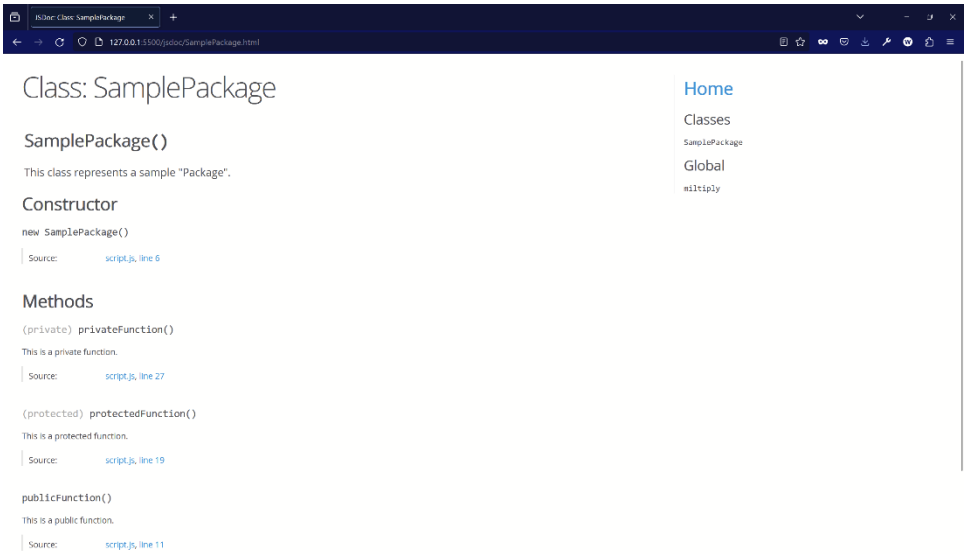

```
jsdoc [source files] -d [doc destination] -a 'all'
```

```
C:\Dydaktyka\TI\Projekt\js>jsdoc . -d ../jsdoc -a 'all'
```

2. Po wykonaniu polecenia zostanie wygenerowana dokumentacja.



Która będzie wyglądać następująco:



Listę dostępnych opcji można uzyskać wpisując polecenie:

jsdoc -h

```
C:\Users\wgalk>jsdoc -h
JSDoc 4.0.2 (Sun, 19 Feb 2023 23:01:18 GMT)

Options:
    -a, --access <value>         Only display symbols with the given access: "package", public", "protected",
                                 "private" or "undefined", or "all" for all access levels. Default: all except
                                 "private"
    -c, --configure <value>      The path to the configuration file. Default: path/to/jsdoc/conf.json
    -d, --destination <value>    The path to the output folder. Default: ./out/
    --debug                      Log information for debugging JSDoc.
    -e, --encoding <value>       Assume this encoding when reading all source files. Default: utf8
    -h, --help                   Print this message and quit.
    --match <value>              When running tests, only use specs whose names contain <value>.
    --nocolor                    When running tests, do not use color in console output.
    -p, --private                Display symbols marked with the @private tag. Equivalent to "--access all".
                                 Default: false
    -P, --package <value>        The path to the project's package file. Default: path/to/sourcefiles/package.json
    --pedantic                   Treat errors as fatal errors, and treat warnings as errors. Default: false
    -q, --query <value>          A query string to parse and store in jsdoc.env.opts.query. Example:
                                 foo=bar&baz=true
    -r, --recurse                Recurse into subdirectories when scanning for source files and tutorials.
    -R, --readme <value>         The path to the project's README file. Default: path/to/sourcefiles/README.md
    -t, --template <value>       The path to the template to use. Default: path/to/jsdoc/templates/default
    -T, --test                   Run all tests and quit.
    -u, --tutorials <value>      Directory in which JSDoc should search for tutorials.
    -v, --version                Display the version number and quit.
    --verbose                    Log detailed information to the console as JSDoc runs.
    -X, --explain                Dump all found doclet internals to console and quit.

Visit https://jsdoc.app/ for more information.
```