

Emilia Wąz 260797

Laboratorium 1.

Lines of code	cloc plato Code Climate
Churn and Complexity	complexity-report Code Climate
Duplication	Code Climate
Linter	plato ESLint
Visualization	plato Code Climate

Maintainability:

Łatwość utrzymania. Wartość proporcjonalna do ilości czasu potrzebnego developerowi/programiście żeby zmienić fragment kodu wraz z ryzykiem że zmiana ta może zaburzyć działanie innego fragmentu kodu.

Objawem braku maintainability może być praca nad zmianą w projekcie, która została oszacowana na czas wielokrotnie mniejszy niż rzeczywisty czas pracy nad nią.



CodeClimate

Maintainability to oszacowanie technicznego długu w repozytorium opartej na standardowej 10-punktowej ocenie duplikacji, złożoności cyklicznej, złożoności poznawczej i kwestii strukturalnych.

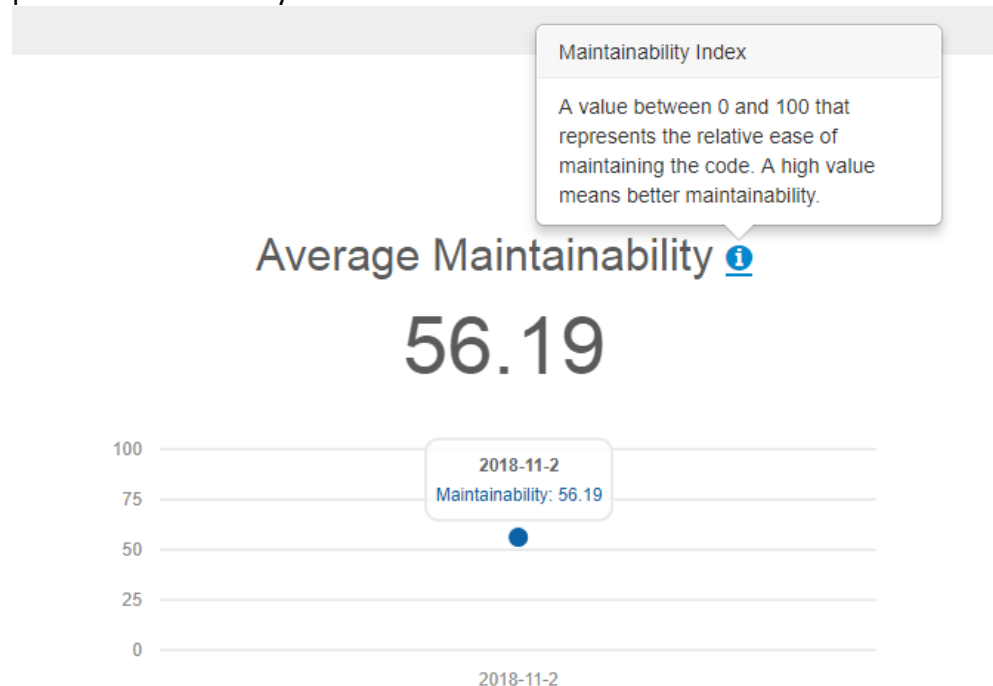
Przypisujemy ocenę literową od A do F w oparciu o całkowity czas naprawy wszystkich zidentyfikowanych problemów z zadłużeniem technicznym.

howler.js

Code Climate: Maintainability F, < 20%

NAME	LINES OF CODE	MAINTAINABILITY ^
 lab1/howler.core.js	1,429	 1 wk

plato: Maintainability 56.19



Churn:

Miara ilości modyfikacji jednego pliku kodu, obejmująca edycję, dodawanie i usuwanie z pliku, czyli zmiany wersji.

Ilość napisanego kodu nie oznacza produktywnego kodowania, a może zawierać wiele modyfikacji istniejącego kodu, czyli właśnie churn.

CodeClimate

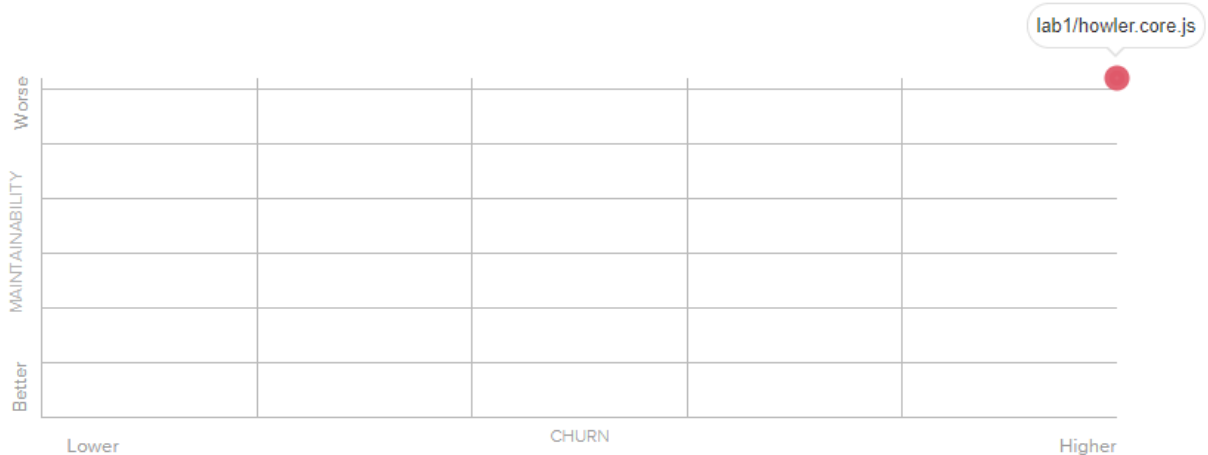
To w przybliżeniu wartość modyfikacji pliku w ciągu ostatnich 90 dni, obliczona przez zliczenie liczby różnych wersji tego pliku w różnych commitach. Problemy z jakością mogą zdarzać się częściej w plikach o wysokim poziomie churn, z tego względu pliki te powinny zwrócić naszą uwagę.

howler.js

High churn

Churn vs. maintainability

Maintainability issues cause bigger problems in files that are changed (churn) frequently.



Duplication:

Powielanie kodu. Kod jest zduplikowany, gdy ogólna struktura jest taka sama, poszczególne operacje i wartości mogą być różne. Przyczyną może być praca kilku programistów nad różnymi fragmentami programu i nieświadomość istnienia duplikatu.

Trudnym do wykrycia duplikatem jest kod, który wykonuje to samo zadanie, ale posiada zupełnie inną strukturę.

Przykładowe rozwiązania:

- kiedy duplikaty znajdują się w różnych metodach tej samej klasy, możliwym rozwiązaniem jest przeniesienie fragmentów do jednej metody lub funkcji.

- kiedy duplikaty znajdują się w różnych klasach pochodnych, metodę, lub fragment konstruktora można stworzyć w klasie bazowej.

- jeśli duplikaty są podobne, ale nie takie same, możliwe jest wydobycie tylko powtarzających się fragmentów i pozostawienie reszty.

CodeClimate

Pliki źródłowe są parsowane na drzewa składniowe. Węzły mają taką samą strukturę, jeśli mają taki sam typ i wszystkie ich podwęzły również mają taką samą strukturę.

Węzły z taką samą strukturą, ale innymi wartościami są klasyfikowane jako podobne.

Węzły z taką samą strukturą i takimi samymi wartościami są klasyfikowane jako identyczne.

howler.js

Code Climate: 16 Duplications

Repository stats

CODE SMELLS

36

DUPLICATION

16

Lines of code:

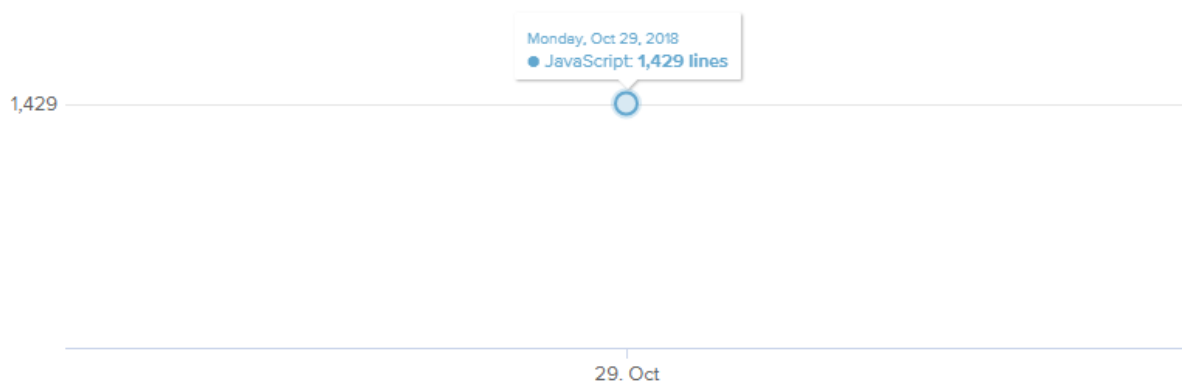
Licznik linii kodu w jednym pliku źródłowym.

howler.js

Code Climate: 1429 lines of code

Lines of code (LOC)

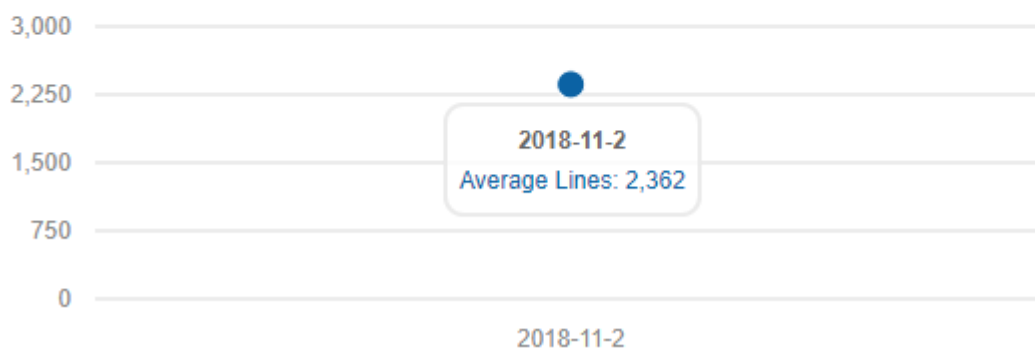
JAVASCRIPT



plato: 2362 total lines of code

Total/Average Lines ⓘ

2362 / 2362



cloc: 1429 lines of code

```
$ cloc src/howler.core.js
  1 text file.
  1 unique file.
  0 files ignored.
```

github.com/AlDanial/cloc v 1.78 T=0.05 s (19.8 files/s, 46766.6 lines/s)

Language	files	blank	comment	code
JavaScript	1	371	562	1429

Lint:

Lint-Program dla języka programowania, który automatycznie sprawdza najczęściej popełniane proste błędy.

Code formatter-Program automatycznie układający kod tak, aby stał się czytelny.

howler.js

plato: 5 errors

Lint errors



Line 213 Column: 13 "'test' is already defined."

```
// Test to make sure audio isn't disabled in Internet Explorer.  
try {  
  var test = new Audio();
```

Column: 13 "'test' is already defined."

```
  if (test.muted) {  
    self.noAudio = true;  
  }  
} catch (e) {}
```

Line 750 Column: 122 "Expected an assignment or function call and instead saw an expression."

```
// Play the sound using the supported method.  
if (typeof node.bufferSource.start === 'undefined') {  
  sound._loop ? node.bufferSource.noteGrainOn(0, seek, 86400) : node.bufferSource.noteGrainOn(0, seek, duration);
```

Column: 122 "Expected an assignment or function call and instead saw an expression."

Line 752 Column: 110 "Expected an assignment or function call and instead saw an expression."

```
  } else {  
    sound._loop ? node.bufferSource.start(0, seek, 86400) : node.bufferSource.start(0, seek, duration);
```

Column: 110 "Expected an assignment or function call and instead saw an expression."

```
  }
```

Line 1740 Column: 22 "Functions declared within loops referencing an outer scoped variable may lead to confusing semantics. (id, msg)"

```
// Loop through event store and fire all functions.  
for (var i=events.length-1; i>=0; i--) {  
  // Only fire the listener if the correct ID is used.  
  if (!events[i].id || events[i].id === id || event === 'load') {  
    setTimeout(function(fn) {
```

Column: 22 "Functions declared within loops referencing an outer scoped variable may lead to confusing semantics. (id, msg)"

Line 2263 Column: 4 "Missing semicolon."

```
/**
 * Decode audio data from an array buffer.
 * @param {ArrayBuffer} arraybuffer The audio data.
 * @param {Howl} self
 */
var decodeAudioData = function(arraybuffer, self) {
  // Fire a load_error if something broke.
  var error = function() {
    self._emit('loaderror', null, 'Decoding audio data failed.');
```

};

// Load the sound on success.

var success = function(buffer) {
 if (buffer && self._sounds.length > 0) {
 cache[self._src] = buffer;
 loadSound(self, buffer);
 } else {
 error();
 }
};

// Decode the buffer into an audio source.

if (typeof Promise !== 'undefined' && Howler.ctx.decodeAudioData.length === 1) {
 Howler.ctx.decodeAudioData(arraybuffer).then(success).catch(error);
} else {
 Howler.ctx.decodeAudioData(arraybuffer, success, error);
}

Column: 4 "Missing semicolon."

Cyclomatic Complexity:

Złożoność cykliczna, czasami określana jako złożoność McCabe'a, jest zliczaniem liniowo niezależnych ścieżek za pomocą kodu źródłowego. "liczba decyzji, które musi wykonać dany blok kodu".

Większość języków zapewnia konstrukcje takie jak if, while itd. dla punktów "decyzji".

complexity-report skala:

- metody 1-10 proste do zrozumienia
- metody 10-20 bardziej złożony kod, testowanie jest trudniejsze
- metody powyżej 20 są typowe dla kodu z dużą ilością potencjalnych ścieżek wykonania, może być w pełni zrozumiany i przetestowany tylko z dużą trudnością i wysiłkiem
- metody powyżej 50 są uznawane za niemożliwe do utrzymania

Cyclomatic complexity density = Cyclomatic complexity / Lines of code

howler.js

complexity-report

```
Function: init
Line No.: 464
Physical LOC: 74
Logical LOC: 59
Parameter count: 1
Cyclomatic complexity: 30
Cyclomatic complexity density: 50.847457627118644%
```

```
Function: play
Line No.: 628
Physical LOC: 232
Logical LOC: 69
Parameter count: 2
Cyclomatic complexity: 23
Cyclomatic complexity density: 33.33333333333333%
```

```
Function: volume
Line No.: 1065
Physical LOC: 73
Logical LOC: 45
Parameter count: 0
Cyclomatic complexity: 17
Cyclomatic complexity density: 37.77777777777778%
```

```
Function: rate
Line No.: 1340
Physical LOC: 84
Logical LOC: 51
Parameter count: 0
Cyclomatic complexity: 17
Cyclomatic complexity density: 33.33333333333333%
```

```
Function: seek
Line No.: 1433
Physical LOC: 98
Logical LOC: 52
Parameter count: 0
```

cyclomatic complexity: 17
cyclomatic complexity density: 32.69230769230769%

Cognitive Complexity:

Złożoność poznawcza jest miarą tego, jak trudno jest intuicyjnie zrozumieć jednostkę kodu. W przeciwieństwie do złożoności cyklicznej, która określa, jak trudny będzie kod do przetestowania, złożoność poznawcza pokazuje, jak trudno będzie odczytać i zrozumieć kod.

Code smells:

Cechy kodu źródłowego mówiące o złym sposobie implementacji i będące sygnałem do refaktoryzacji.

howler.js

CodeClimate: 36 code smells

Repository stats

CODE SMELLS

36

DUPLICATION

16

1. Komunikaty związane z duplikatami, przykład:

```
1092         if (self._state !== 'loaded' || self._playLock) {
1093             self._queue.push({
1094                 event: 'volume',
1095                 action: function() {
1096                     self.volume.apply(self, args);
```

Found in [lab1/howler.core.js](#) and 2 other locations - About 2 hrs to fix

Similar blocks of code found in 3 locations. Consider refactoring.

OPEN

```
1463         if (self._state !== 'loaded' || self._playLock) {
1464             self._queue.push({
1465                 event: 'seek',
1466                 action: function() {
1467                     self.seek.apply(self, args);
```

Found in [lab1/howler.core.js](#) and 2 other locations - About 2 hrs to fix

Similar blocks of code found in 3 locations. Consider refactoring.

OPEN

```
1367         if (self._state !== 'loaded' || self._playLock) {
1368             self._queue.push({
1369                 event: 'rate',
1370                 action: function() {
1371                     self.rate.apply(self, args);
```

Found in [lab1/howler.core.js](#) and 2 other locations - About 2 hrs to fix

Rozwiązanie:

Powtarzające się bloki kodu wyekstraktować i stworzyć jedną funkcję lub metodę.

```

/* FROM */
if (self._state !== 'loaded' || self._playLock) {
  self._queue.push({
    event: 'rate',
    action: function() {
      self.rate.apply(self, args);
    }
  });
}
return self;
}

/* TO */
if (self._state !== 'loaded' || self._playLock) {
  this.selfFunction(self, args);
  return self;
}

selfFunction: function(self, args, eventName) {
  self._queue.push({
    event: eventName,
    action: function() {
      self.rate.apply(self, args);
    }
  });
}
}

```

2. Głęboko zagnieżdżone instrukcje przepływu sterowania.

Rozwiązanie:

Wyekstraktować bloki warunkowe w oddzielne funkcje.

Przekonwertować warunki kontrolne negatywne na pozytywne.

Powrót z funkcji tak szybko, jak to możliwe.

```

/* FROM */
if (sound._node) {
    if (self._webAudio) {
        // Make sure the sound has been created.
        if (!sound._node.bufferSource) {
            continue;
        }
        if (typeof sound._node.bufferSource.stop === 'undefined') {
            sound._node.bufferSource.noteOff(0);
        } else {
            sound._node.bufferSource.stop(0);
        }
        // Clean up the buffer source.
        self._cleanBuffer(sound._node);
    } else if (!isNaN(sound._node.duration) || sound._node.duration === Infinity) {
        sound._node.pause();
    }
}

/* TO */
var check = this.checkFunction1(),
check = this.checkFunction1() && this.checkFunction2(),
check = this.checkFunction1() && this.checkFunction2() && this.checkFunction3(),
check = this.checkFunction1() && this.checkFunction2() && this.checkFunction4(),
check = this.checkFunction1() && this.checkFunction5();

checkFunction1: function() {
    var valid = sound._node;

    if(valid) {
        // Reset the seek position.
        sound._seek = self.seek(ids[i]);
        sound._rateSeek = 0;
        sound._paused = true;
        // Stop currently running fades.
        self._stopFade(ids[i]);

        return true;
    }
    return false;
}
}

```

3. Zbyt wiele argumentów funkcji, przykład:

Function `_startFadeInterval` has 6 arguments (exceeds 4 allowed).

Consider refactoring.

[OPEN](#)

```
1208      _startFadeInterval: function(sound, from, to, len, id, isGroup) {
```

Found in `src/howler.core.js` - About 45 mins to fix

Rozwiązanie:

Zamiana zmiennych na obiekt zawierający potrzebne elementy i przekazanie do funkcji tylko jednego argumentu.

```
/* FROM */
_startFadeInterval: function(sound, from, to, len, id, isGroup) {}

/* TO */
interval = {
  from: Number,
  to: Number,
  len: Number,
  id: Number,
  isGroup: Boolean
}

_startFadeInterval: function(sound, interval) {}
```

4. Skomplikowane wyrażenie logiczne, przykład:

Consider simplifying this complex logical expression.

[OPEN](#)

```
2318      if (Howler._navigator && Howler._navigator.standalone && !safari || Howler._navigator &&
2319          Howler.usingWebAudio = false;
2320      }
```

Found in `src/howler.core.js` - About 40 mins to fix

`Howler._navigator && Howler._navigator.standalone && !safari || Howler._navigator && !Howler._navigator.standalone && !safari`

Rozwiązanie:

Użycie kalkulatora wyrażeń logicznych, aby uprościć wyrażenie.

```
/* FROM */  
if(Howler._navigator && Howler._navigator.standalone && !safari  
  || Howler._navigator && !Howler._navigator.standalone && !safari)  
/* TO */  
if(Howler._navigator && !safari)
```