TC39 스펙에 대한 주관적 참견 시점서 재원(@ENvironmentSet)

Making Programming Lanugage

Standard

JavaScript

- 넷스케이프의 엔지니어 ‘브랜던 아이크’
- Mocha → LiveScript → JavaScript
- 넷스케이프 네비게이터 2.0 베타에서 공개

JScript

- 마이크로소프트
- JavaScript → Jscript
- 인터넷 익스플로러 3.0 에서 공개

ECMAScript

- ECMA 재단 ‘TC(Technical Committee) 39’
- JavaScript + Jscript → ECMAScript
- ECMA-262 1판에서 공개

Proposal

The TC39 Process

- Stage 0, Strawman – 아이디어
- Stage 1, Proposal - 제안서
- Stage 2, Draft - 불완전한 명세
- Stage 3, Candidate - 완전한 명세
- Stage 4, Finished - 표준

Contribute

컨트리뷰트 할 대상 찾기

- TC39 gihub (github.com/tc39)
- tc39/ecma262 (현재 표준)
- tc39/proposals (proposal들)

tc39/ecma262

- 최신 자바스크립트 표준 문서가 보관되있는 저장소
- 현재 표준에 있는 문제점이 주 이슈

tc39/proposals

- tc39에 제출된 모든 proposal 들이 보관되있는 저장소
- 모든 proposal 들의 진행 사항을 한 눈에 볼 수 있다

proposal-xxxxxx

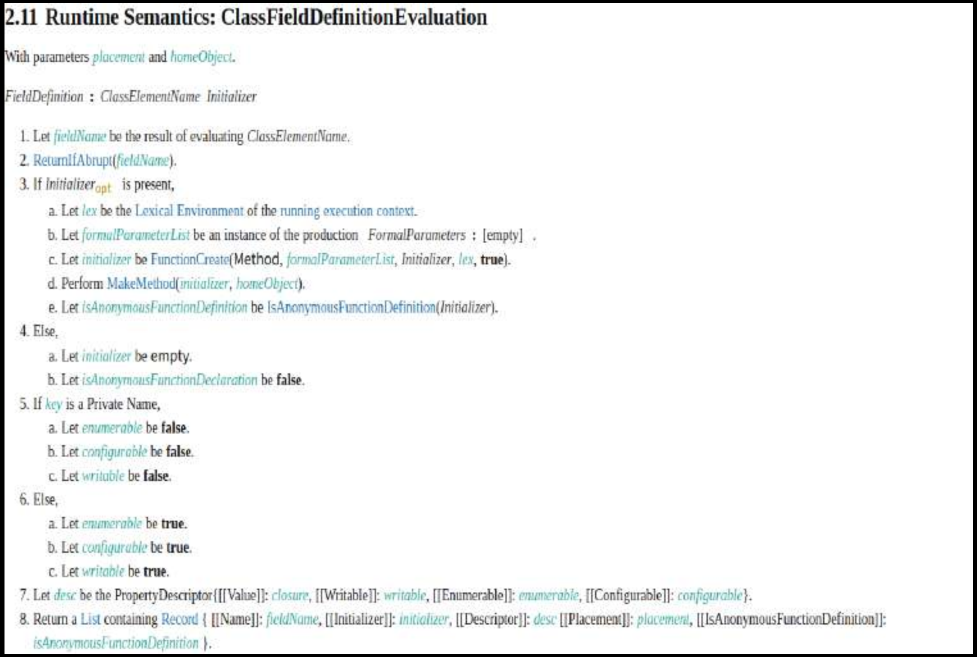
- tc39에 제출된 proposal의 전용 저장소
- 문제점, 추가 기능 등이 주 된 이슈

decorator

proposal-decorators

- 자바스크립트에 데코레이터를 추가하는 proposal
- 다른 여러 proposal 들에 기반하여 제작되었다
- Class fields(클래스의 필드를 정의하는 문법 추가)
- Private methods(클래스의 private 메서드를 정의하는 문법 추가)
- Orthogonal Classes(클래스와 관련된 좋은 문법들을 만드는 방법)

ClassFieldDefinitionEvaluation



2.11 Runtime Semantics: ClassFieldDefinitionEvaluation

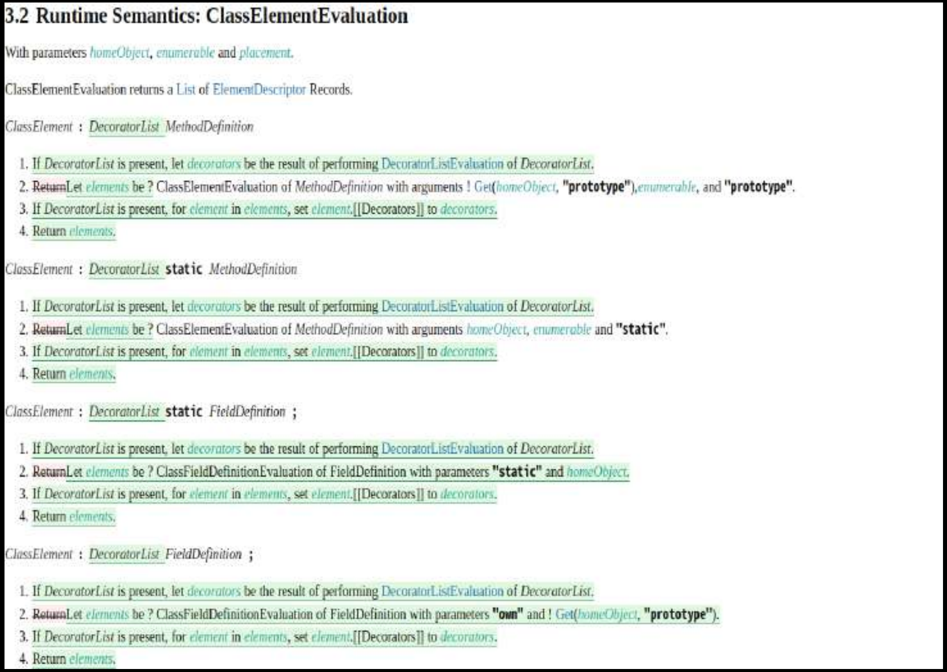
With parameters *placement* and *homeObject*.

FieldDefinition : *ClassElementName* *Initializer*

```
1. Let fieldName be the result of evaluating ClassElementName.
2. ReturnIfAbrupt(fieldName).
3. If Initializer_opt is present,
    a. Let lex be the Lexical Environment of the running execution context.
    b. Let formalParameterList be an instance of the production  FormalParameters : [empty]  .
    c. Let initializer be FunctionCreate(Method, formalParameterList, Initializer, lex, true).
    d. Perform MakeMethod(initializer, homeObject).
    e. Let isAnonymousFunctionDefinition be IsAnonymousFunctionDefinition(Initializer).
4. Else,
    a. Let initializer be empty.
    b. Let isAnonymousFunctionDeclaration be false.
5. If key is a Private Name,
    a. Let enumerable be false.
    b. Let configurable be false.
    c. Let writable be false.
6. Else,
    a. Let enumerable be true.
    b. Let configurable be true.
    c. Let writable be true.
7. Let desc be the PropertyDescriptor{[[Value]]: closure, [[Writable]]: writable, [[Enumerable]]: enumerable, [[Configurable]]: configurable}.
8. Return a List containing Record { [[Name]]: fieldName, [[Initializer]]: initializer, [[Descriptor]]: desc [[Placement]]: placement, [[IsAnonymousFunctionDefinition]]:
   isAnonymousFunctionDefinition }.
```

- 클래스 필드 선언을 평가하는 서브루틴
- 평가된 클래스 필드의 정보들을 모아 둔 객체를 반환한다.

ClassElementEvaluation



3.2 Runtime Semantics: ClassElementEvaluation

With parameters *homeObject*, *enumerable* and *placement*.

ClassElementEvaluation returns a List of ElementDescriptor Records.

ClassElement : *DecoratorList* *MethodDefinition*

1. If *DecoratorList* is present, let *decorators* be the result of performing DecoratorListEvaluation of *DecoratorList*.
2. ~~Return~~Let *elements* be ? ClassElementEvaluation of *MethodDefinition* with arguments ! Get(*homeObject*, **"prototype"**),*enumerable*, and **"prototype"**.
3. If *DecoratorList* is present, for *element* in *elements*, set *element*.[[Decorators]] to *decorators*.
4. Return *elements*.

ClassElement : *DecoratorList* **static** *MethodDefinition*

1. If *DecoratorList* is present, let *decorators* be the result of performing DecoratorListEvaluation of *DecoratorList*.
2. ~~Return~~Let *elements* be ? ClassElementEvaluation of *MethodDefinition* with arguments *homeObject*, *enumerable* and **"static"**.
3. If *DecoratorList* is present, for *element* in *elements*, set *element*.[[Decorators]] to *decorators*.
4. Return *elements*.

ClassElement : *DecoratorList* **static** *FieldDefinition* ;

1. If *DecoratorList* is present, let *decorators* be the result of performing DecoratorListEvaluation of *DecoratorList*.
2. ~~Return~~Let *elements* be ? ClassFieldDefinitionEvaluation of FieldDefinition with parameters **"static"** and *homeObject*.
3. If *DecoratorList* is present, for *element* in *elements*, set *element*.[[Decorators]] to *decorators*.
4. Return *elements*.

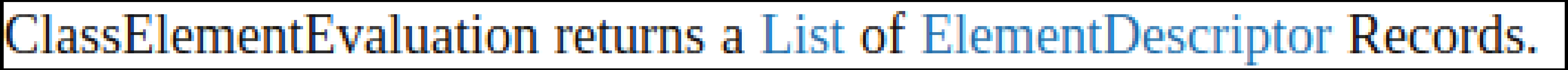
ClassElement : *DecoratorList* *FieldDefinition* ;

1. If *DecoratorList* is present, let *decorators* be the result of performing DecoratorListEvaluation of *DecoratorList*.
2. ~~Return~~Let *elements* be ? ClassFieldDefinitionEvaluation of FieldDefinition with parameters **"own"** and ! Get(*homeObject*, **"prototype"**).
3. If *DecoratorList* is present, for *element* in *elements*, set *element*.[[Decorators]] to *decorators*.
4. Return *elements*.

- 클래스의 엘리먼트 선언을 평가하는 서브루틴
- ClassFieldDefintionEvaluation 을 호출하고 그 결과를 조금 가공해서 반환하고 있다.

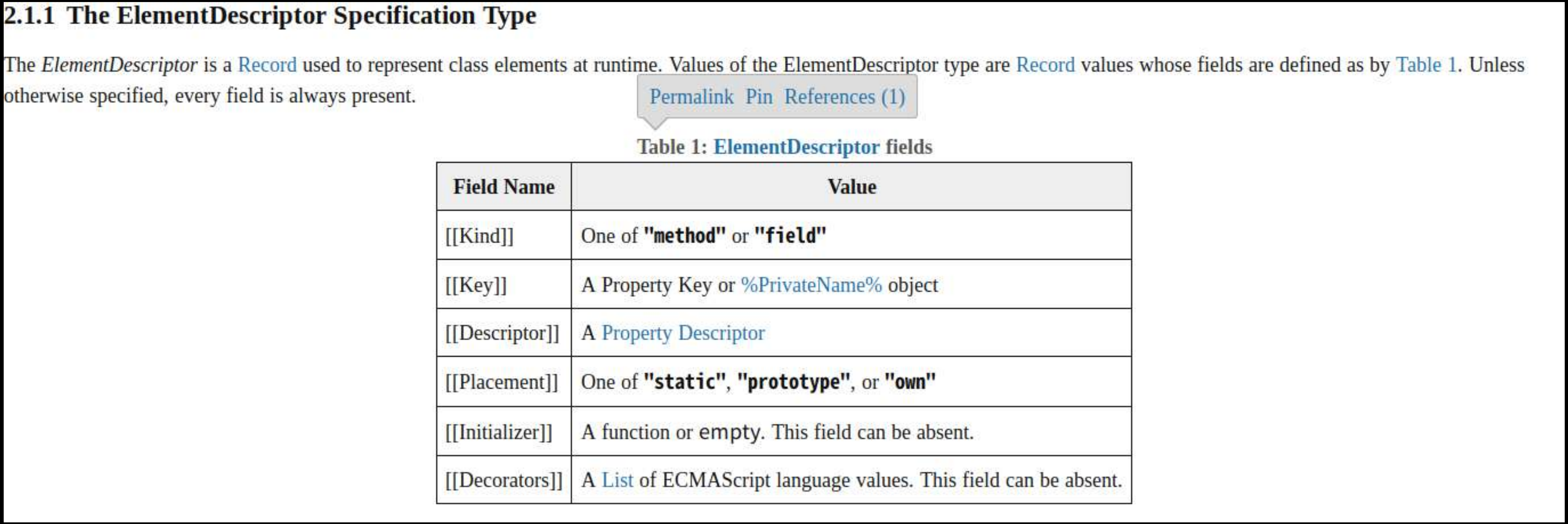
그래서 뭐가 문제죠?

다시 살펴봅시다



ClassElementEvaluation returns a List of ElementDescriptor Records.

ElementDescriptor



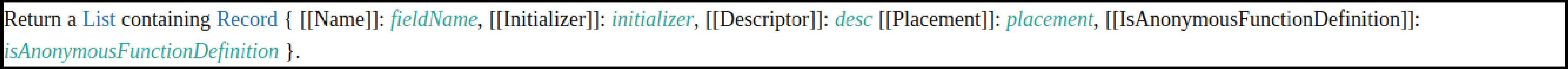
2.1.1 The ElementDescriptor Specification Type

The *ElementDescriptor* is a Record used to represent class elements at runtime. Values of the ElementDescriptor type are Record values whose fields are defined as by Table 1. Unless otherwise specified, every field is always present.

Table 1: ElementDescriptor fields

Field Name	Value
[[Kind]]	One of **"method"** or **"field"**
[[Key]]	A Property Key or %PrivateName% object
[[Descriptor]]	A Property Descriptor
[[Placement]]	One of **"static"**, **"prototype"**, or **"own"**
[[Initializer]]	A function or empty. This field can be absent.
[[Decorators]]	A List of ECMAScript language values. This field can be absent.

ClassFieldDefinitionEvaluation



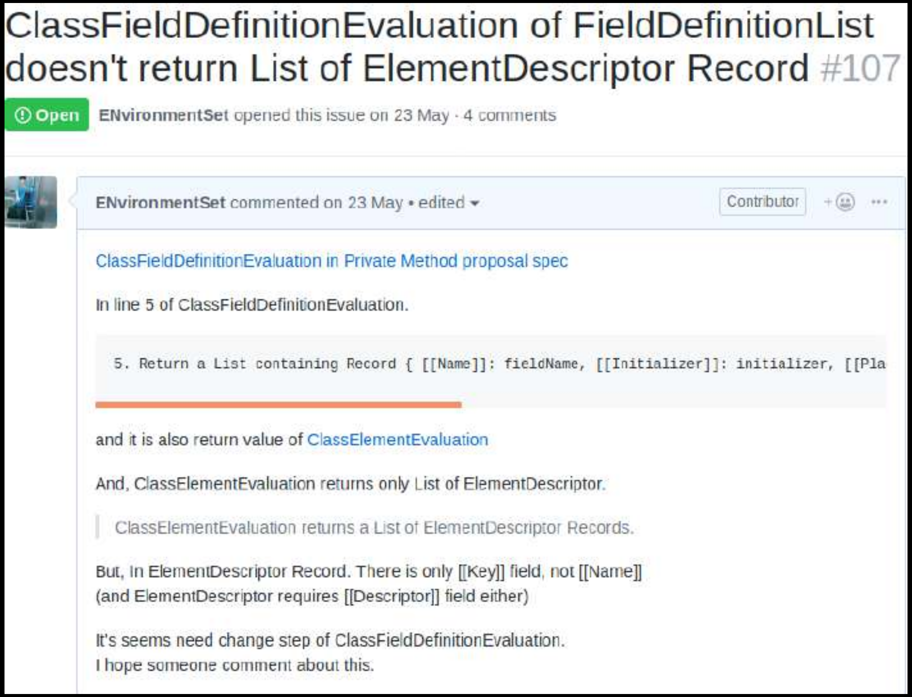
Return a List containing Record { [[Name]]: *fieldName*, [[Initializer]]: *initializer*, [[Descriptor]]: *desc* [[Placement]]: *placement*, [[IsAnonymousFunctionDefinition]]: *isAnonymousFunctionDefinition* }.

엇갈린 서브루틴

- ClassElementEvaluation 은 ElementDescriptor를 반환한다
- ClassElementEvaluation 은ClassFieldDefinitionEvaluation 의 반환값을 조작하여 사용한다
- ClassFieldDefinitionEvaluation 은 ElementDescriptor를 반환하지 않는다
- 두 서브루틴의 동작이 어긋났다!

문제를 고쳐 봅시다

Issue #107



ClassFieldDefinitionEvaluation of FieldDefinitionList doesn't return List of ElementDescriptor Record #107

Open ENvironmentSet opened this issue on 23 May · 4 comments

ENvironmentSet commented on 23 May • edited

Contributor

ClassFieldDefinitionEvaluation in Private Method proposal spec

In line 5 of ClassFieldDefinitionEvaluation.

```
5. Return a List containing Record { [[Name]]: fieldName, [[Initializer]]: initializer, [[Pla
```

and it is also return value of ClassElementEvaluation

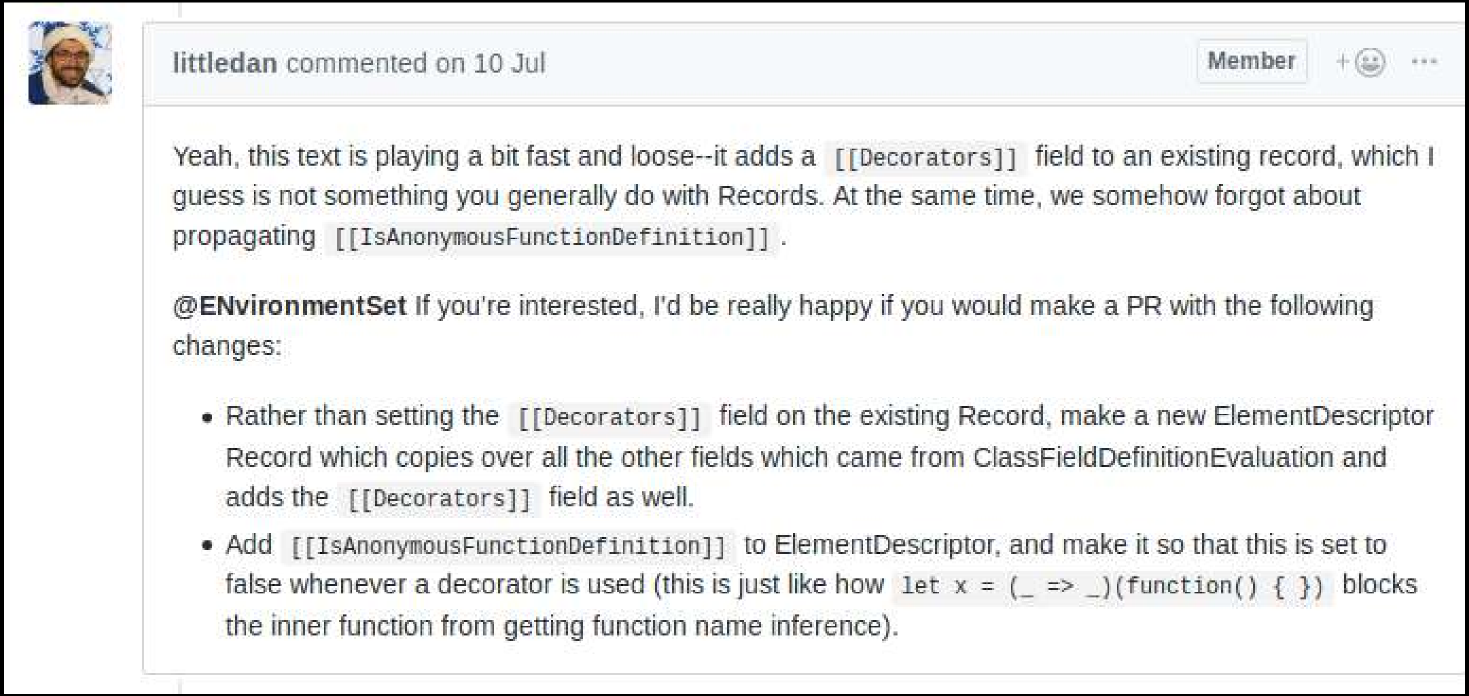
And, ClassElementEvaluation returns only List of ElementDescriptor.

> ClassElementEvaluation returns a List of ElementDescriptor Records.

But, In ElementDescriptor Record. There is only [[Key]] field, not [[Name]]
(and ElementDescriptor requires [[Descriptor]] field either)

It's seems need change step of ClassFieldDefinitionEvaluation.
I hope someone comment about this.

챔피언의 답변



littledan commented on 10 Jul

Member

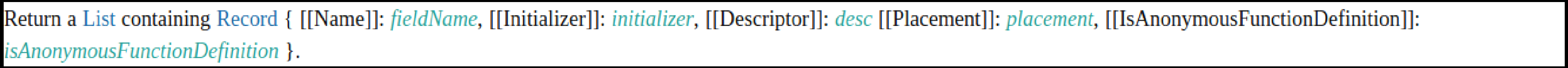
Yeah, this text is playing a bit fast and loose--it adds a `[[Decorators]]` field to an existing record, which I guess is not something you generally do with Records. At the same time, we somehow forgot about propagating `[[IsAnonymousFunctionDefinition]]`.

@ENvironmentSet If you're interested, I'd be really happy if you would make a PR with the following changes:

- Rather than setting the `[[Decorators]]` field on the existing Record, make a new ElementDescriptor Record which copies over all the other fields which came from ClassFieldDefinitionEvaluation and adds the `[[Decorators]]` field as well.
- Add `[[IsAnonymousFunctionDefinition]]` to ElementDescriptor, and make it so that this is set to false whenever a decorator is used (this is just like how `let x = (_ => _)(function() { })` blocks the inner function from getting function name inference).

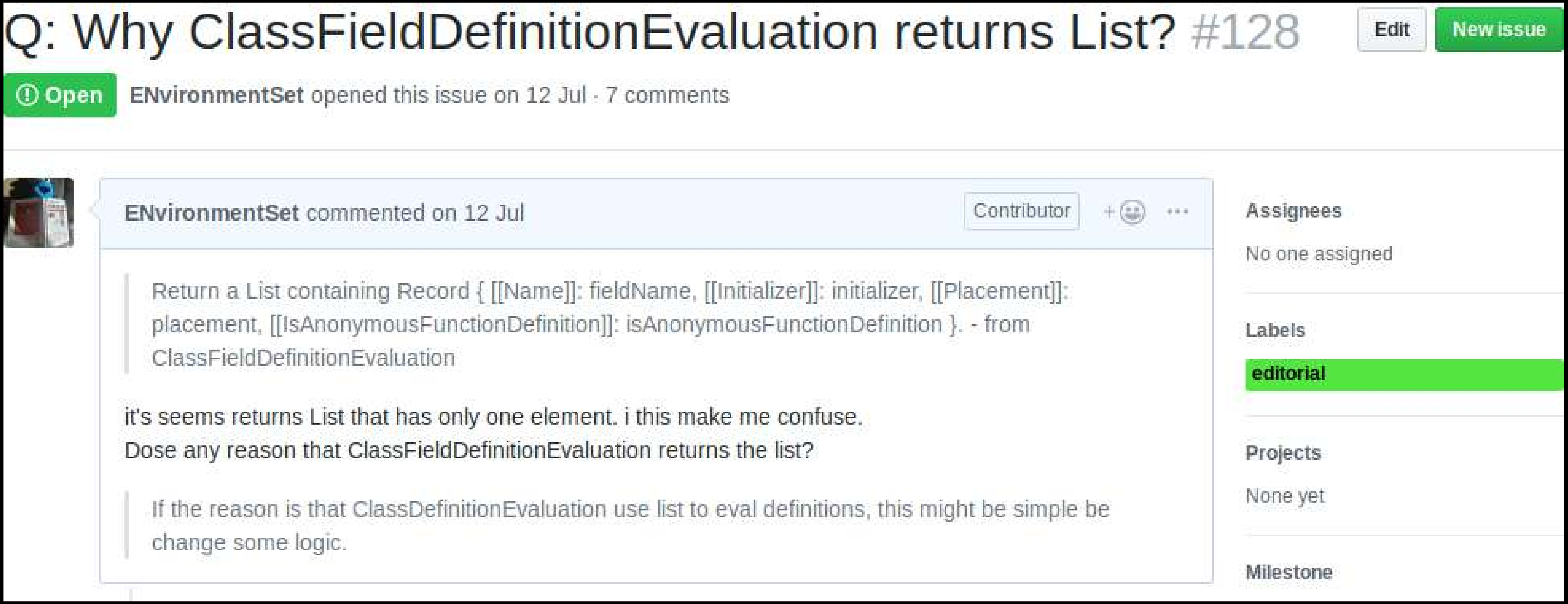
버그의 법칙

ClassFieldDefinitionEvaluation



Return a List containing Record { [[Name]]: *fieldName*, [[Initializer]]: *initializer*, [[Descriptor]]: *desc* [[Placement]]: *placement*, [[IsAnonymousFunctionDefinition]]: *isAnonymousFunctionDefinition* }.

Issue #128



Q: Why ClassFieldDefinitionEvaluation returns List? #128

Open **ENvironmentSet** opened this issue on 12 Jul · 7 comments

ENvironmentSet commented on 12 Jul

> Return a List containing Record { [[Name]]: fieldName, [[Initializer]]: initializer, [[Placement]]: placement, [[IsAnonymousFunctionDefinition]]: isAnonymousFunctionDefinition }. - from ClassFieldDefinitionEvaluation

it's seems returns List that has only one element. i this make me confuse.
Dose any reason that ClassFieldDefinitionEvaluation returns the list?

> If the reason is that ClassDefinitionEvaluation use list to eval definitions, this might be simple be change some logic.

Assignees
No one assigned

Labels
editorial

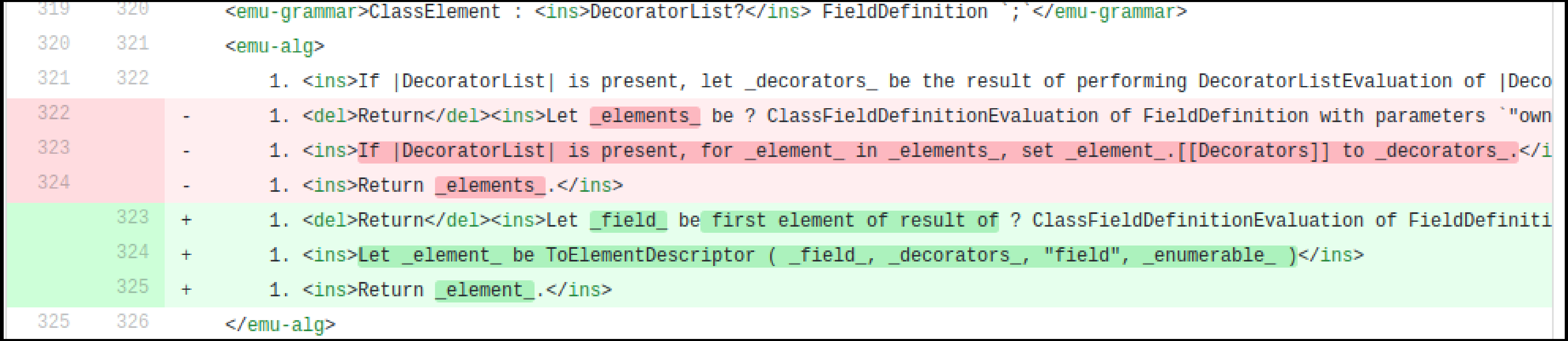
Projects
None yet

Milestone

PR에 들어가야 하는 내용

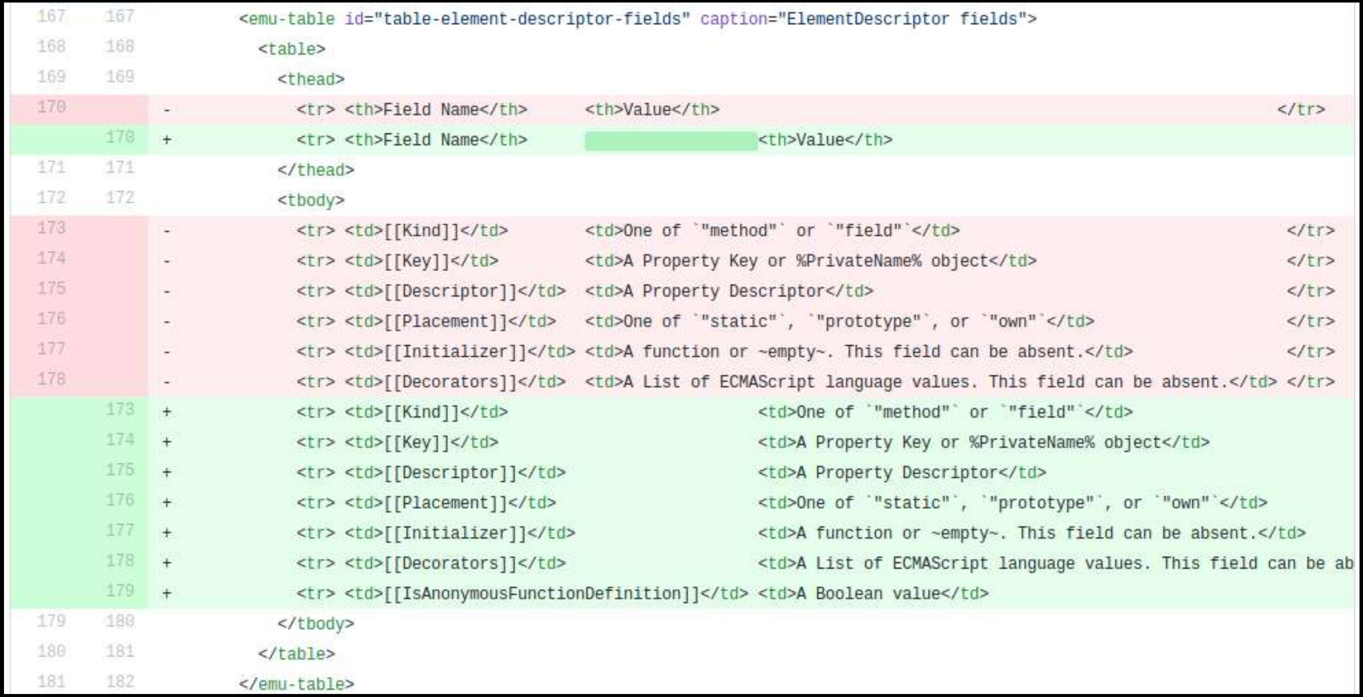
- ClassFieldDefinitionEvaluation이 Record 를 반환하게 변경하기
- ElementDescriptor 에 빠진 [[IsAnonymousFunctionDefinition]] 필드 추가하기
- ClassElementEvaluation과 ClassFieldDefinitionEvaluation 사이의 엇갈린 부분을 해결해 줄 서브루틴 만들기

ClassElementEvaluation 변경하기



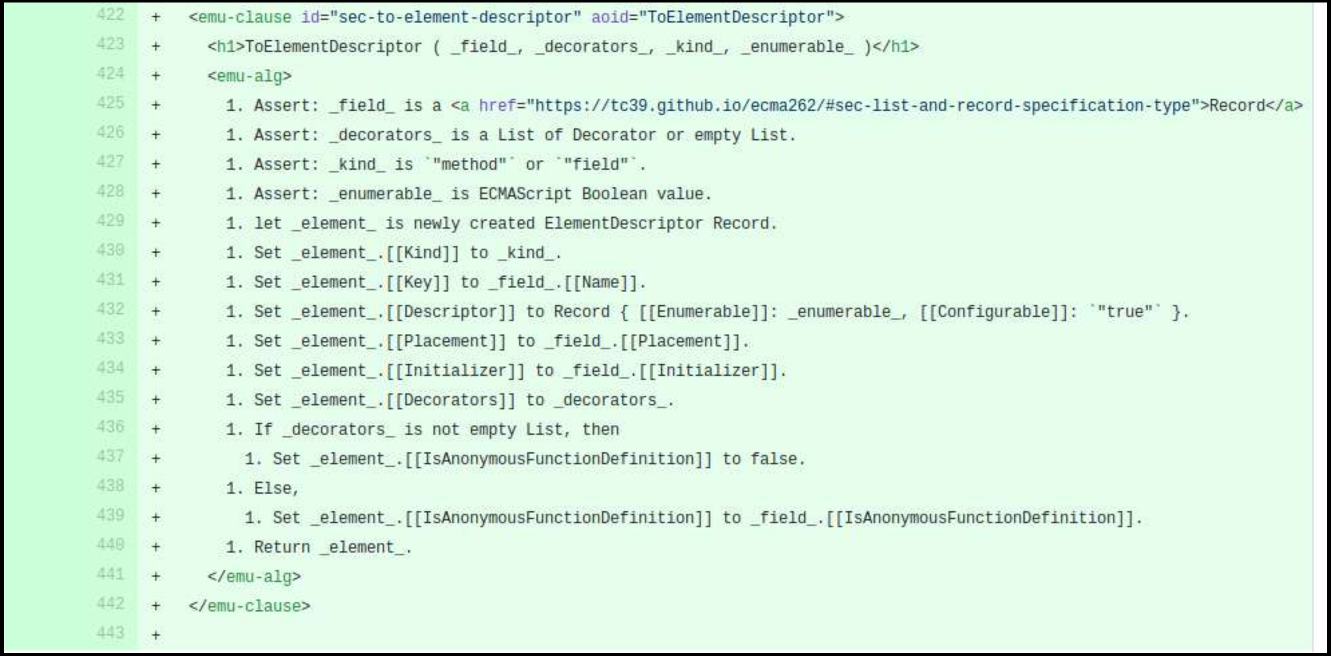
```
  <emu-grammar>ClassElement : <ins>DecoratorList?</ins> FieldDefinition `;`</emu-grammar>
  <emu-alg>
    1. <ins>If |DecoratorList| is present, let _decorators_ be the result of performing DecoratorListEvaluation of |Deco
-   1. <del>Return</del><ins>Let _elements_ be ? ClassFieldDefinitionEvaluation of FieldDefinition with parameters `"own
-   1. <ins>If |DecoratorList| is present, for _element_ in _elements_, set _element_.[[Decorators]] to _decorators_.</i
-   1. <ins>Return _elements_.</ins>
+   1. <del>Return</del><ins>Let _field_ be first element of result of ? ClassFieldDefinitionEvaluation of FieldDefiniti
+   1. <ins>Let _element_ be ToElementDescriptor ( _field_, _decorators_, "field", _enumerable_ )</ins>
+   1. <ins>Return _element_.</ins>
  </emu-alg>
```

[[IsAnonymousFunctionDefinition]] 필드 추가하기



```diff
      <emu-table id="table-element-descriptor-fields" caption="ElementDescriptor fields">
        <table>
          <thead>
-           <tr> <th>Field Name</th>      <th>Value</th>                                                            </tr>
+           <tr> <th>Field Name</th>                         <th>Value</th>
          </thead>
          <tbody>
-           <tr> <td>[[Kind]]</td>        <td>One of `"method"` or `"field"`</td>                                    </tr>
-           <tr> <td>[[Key]]</td>         <td>A Property Key or %PrivateName% object</td>                            </tr>
-           <tr> <td>[[Descriptor]]</td>  <td>A Property Descriptor</td>                                             </tr>
-           <tr> <td>[[Placement]]</td>   <td>One of `"static"`, `"prototype"`, or `"own"`</td>                      </tr>
-           <tr> <td>[[Initializer]]</td> <td>A function or ~empty~. This field can be absent.</td>                  </tr>
-           <tr> <td>[[Decorators]]</td>  <td>A List of ECMAScript language values. This field can be absent.</td> </tr>
+           <tr> <td>[[Kind]]</td>                           <td>One of `"method"` or `"field"`</td>
+           <tr> <td>[[Key]]</td>                            <td>A Property Key or %PrivateName% object</td>
+           <tr> <td>[[Descriptor]]</td>                     <td>A Property Descriptor</td>
+           <tr> <td>[[Placement]]</td>                      <td>One of `"static"`, `"prototype"`, or `"own"`</td>
+           <tr> <td>[[Initializer]]</td>                    <td>A function or ~empty~. This field can be absent.</td>
+           <tr> <td>[[Decorators]]</td>                     <td>A List of ECMAScript language values. This field can be ab
+           <tr> <td>[[IsAnonymousFunctionDefinition]]</td> <td>A Boolean value</td>
          </tbody>
        </table>
      </emu-table>
```

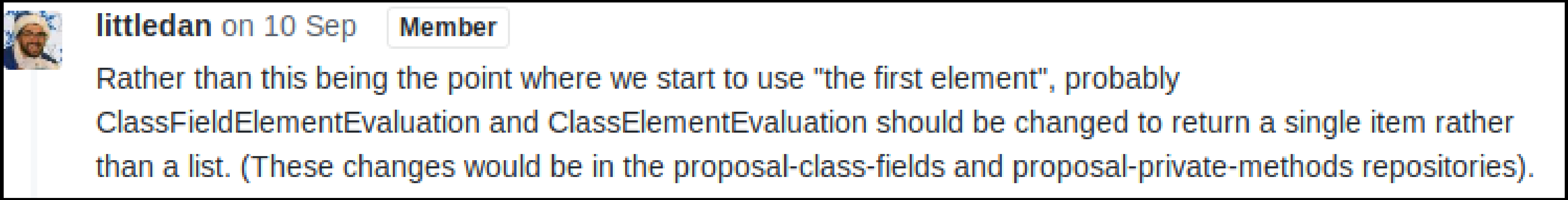
ToElementDescriptor



```
+  <emu-clause id="sec-to-element-descriptor" aoid="ToElementDescriptor">
+    <h1>ToElementDescriptor ( _field_, _decorators_, _kind_, _enumerable_ )</h1>
+    <emu-alg>
+      1. Assert: _field_ is a <a href="https://tc39.github.io/ecma262/#sec-list-and-record-specification-type">Record</a>
+      1. Assert: _decorators_ is a List of Decorator or empty List.
+      1. Assert: _kind_ is `"method"` or `"field"`.
+      1. Assert: _enumerable_ is ECMAScript Boolean value.
+      1. let _element_ is newly created ElementDescriptor Record.
+      1. Set _element_.[[Kind]] to _kind_.
+      1. Set _element_.[[Key]] to _field_.[[Name]].
+      1. Set _element_.[[Descriptor]] to Record { [[Enumerable]]: _enumerable_, [[Configurable]]: `"true"` }.
+      1. Set _element_.[[Placement]] to _field_.[[Placement]].
+      1. Set _element_.[[Initializer]] to _field_.[[Initializer]].
+      1. Set _element_.[[Decorators]] to _decorators_.
+      1. If _decorators_ is not empty List, then
+        1. Set _element_.[[IsAnonymousFunctionDefinition]] to false.
+      1. Else,
+        1. Set _element_.[[IsAnonymousFunctionDefinition]] to _field_.[[IsAnonymousFunctionDefinition]].
+      1. Return _element_.
+    </emu-alg>
+  </emu-clause>
+
```

또 다시 버그의 법칙

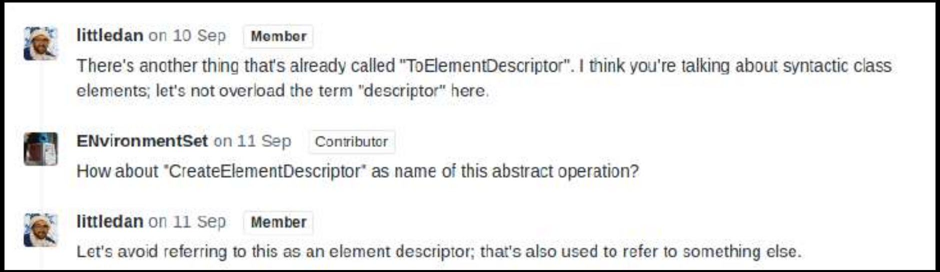
class-field/private-method



littledan on 10 Sep Member

Rather than this being the point where we start to use "the first element", probably ClassFieldElementEvaluation and ClassElementEvaluation should be changed to return a single item rather than a list. (These changes would be in the proposal-class-fields and proposal-private-methods repositories).

겹쳐버린 이름



littledan on 10 Sep Member

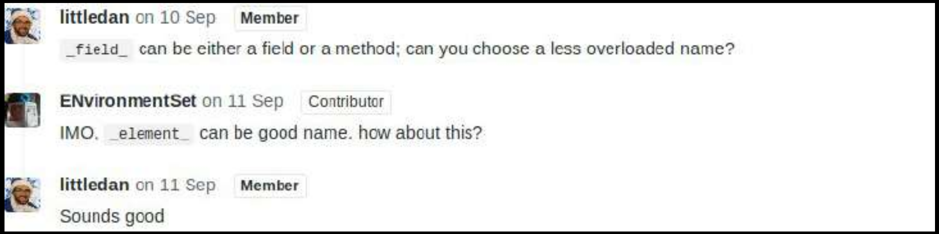
There's another thing that's already called "ToElementDescriptor". I think you're talking about syntactic class elements; let's not overload the term "descriptor" here.

ENvironmentSet on 11 Sep Contributor

How about "CreateElementDescriptor" as name of this abstract operation?

littledan on 11 Sep Member

Let's avoid referring to this as an element descriptor; that's also used to refer to something else.



littledan on 10 Sep Member

`_field_` can be either a field or a method; can you choose a less overloaded name?

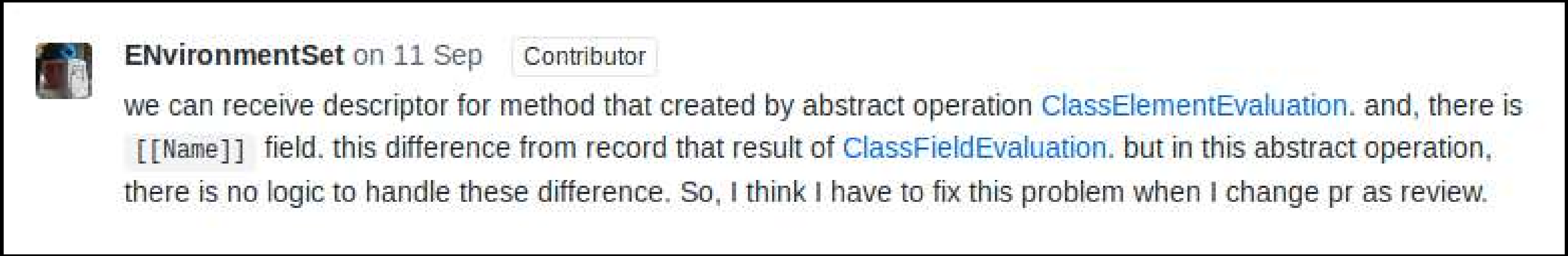
ENvironmentSet on 11 Sep Contributor

IMO, `_element_` can be good name. how about this?

littledan on 11 Sep Member

Sounds good

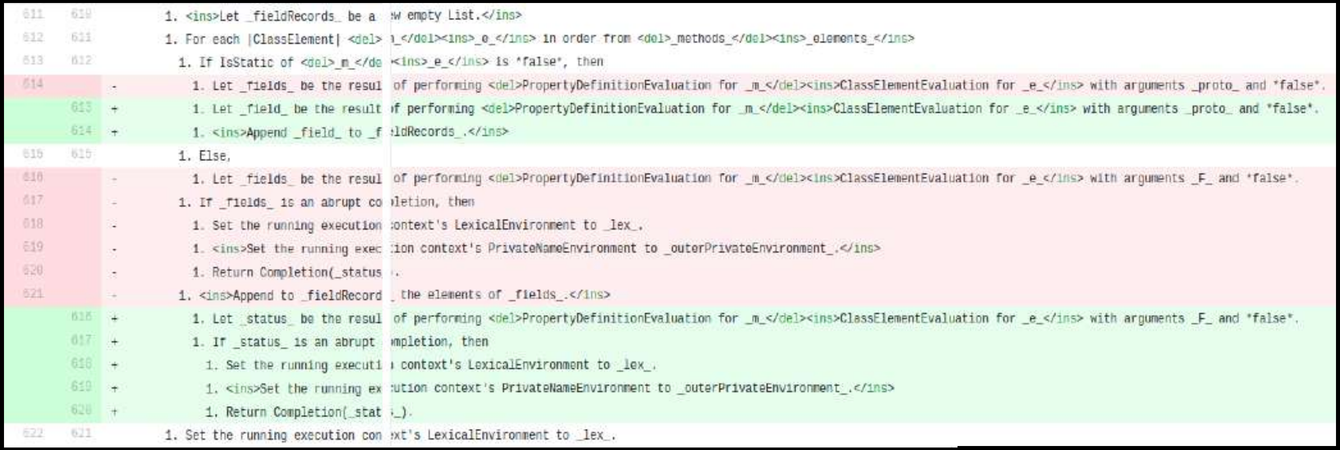
Duck debugging



ENvironmentSet on 11 Sep Contributor

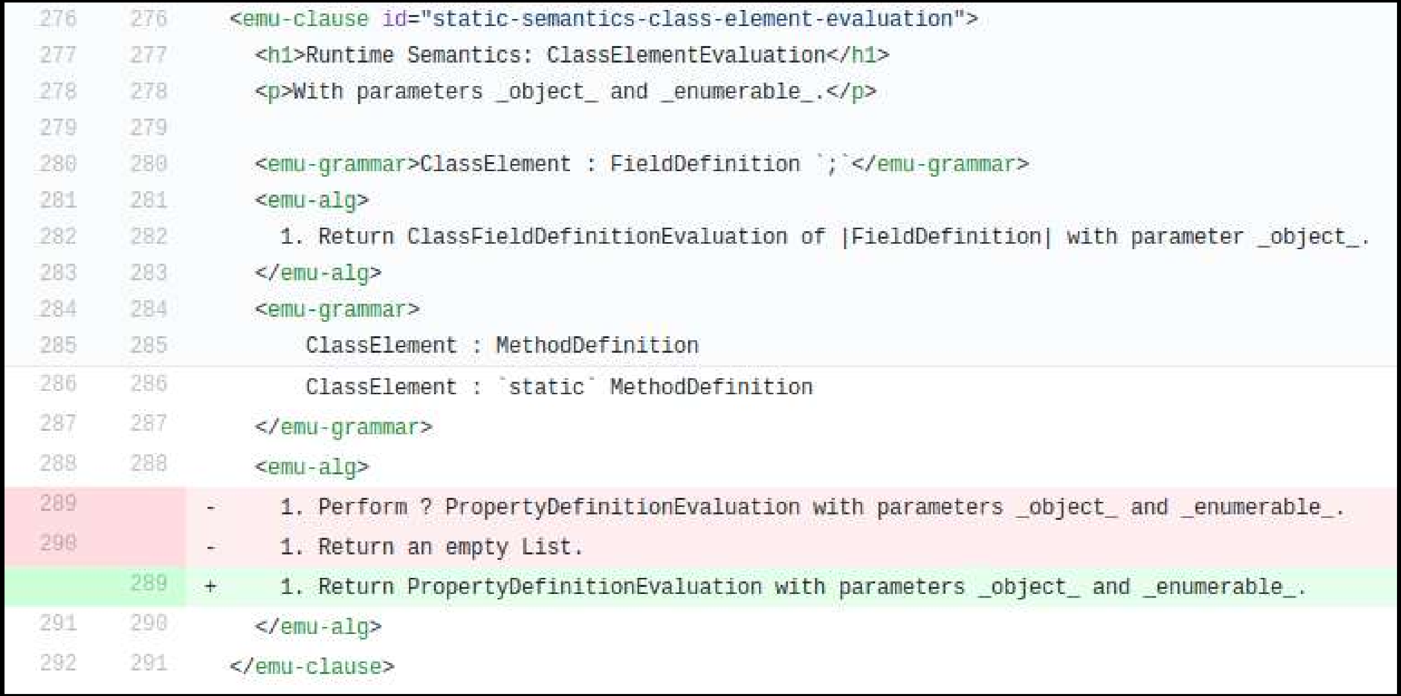
we can receive descriptor for method that created by abstract operation ClassElementEvaluation. and, there is `[[Name]]` field. this difference from record that result of ClassFieldEvaluation. but in this abstract operation, there is no logic to handle these difference. So, I think I have to fix this problem when I change pr as review.

ClassDefinitionEvaluation



```
      1. <ins>Let _fieldRecords_ be a  :w empty List.</ins>
      1. For each |ClassElement| <del> ı_</del><ins>_e_</ins> in order from <del>_methods_</del><ins>_elements_</ins>
        1. If IsStatic of <del>_m_</de <ins>_e_</ins> is *false*, then
-         1. Let _fields_ be the resul  of performing <del>PropertyDefinitionEvaluation for _m_</del><ins>ClassElementEvaluation for _e_</ins> with arguments _proto_ and *false*.
+         1. Let _field_ be the result ıf performing <del>PropertyDefinitionEvaluation for _m_</del><ins>ClassElementEvaluation for _e_</ins> with arguments _proto_ and *false*.
+         1. <ins>Append _field_ to _f :ldRecords_.</ins>
        1. Else,
-         1. Let _fields_ be the resul  of performing <del>PropertyDefinitionEvaluation for _m_</del><ins>ClassElementEvaluation for _e_</ins> with arguments _F_ and *false*.
-       1. If _fields_ is an abrupt co ıletion, then
-         1. Set the running execution :ontext's LexicalEnvironment to _lex_.
-         1. <ins>Set the running exec :ion context's PrivateNameEnvironment to _outerPrivateEnvironment_.</ins>
-         1. Return Completion(_status ı.
-       1. <ins>Append to _fieldRecord _ the elements of _fields_.</ins>
+         1. Let _status_ be the resul  of performing <del>PropertyDefinitionEvaluation for _m_</del><ins>ClassElementEvaluation for _e_</ins> with arguments _F_ and *false*.
+         1. If _status_ is an abrupt  ımpletion, then
+           1. Set the running executi ı context's LexicalEnvironment to _lex_.
+           1. <ins>Set the running ex :ution context's PrivateNameEnvironment to _outerPrivateEnvironment_.</ins>
+           1. Return Completion(_stat ;_).
      1. Set the running execution con :xt's LexicalEnvironment to _lex_.
```

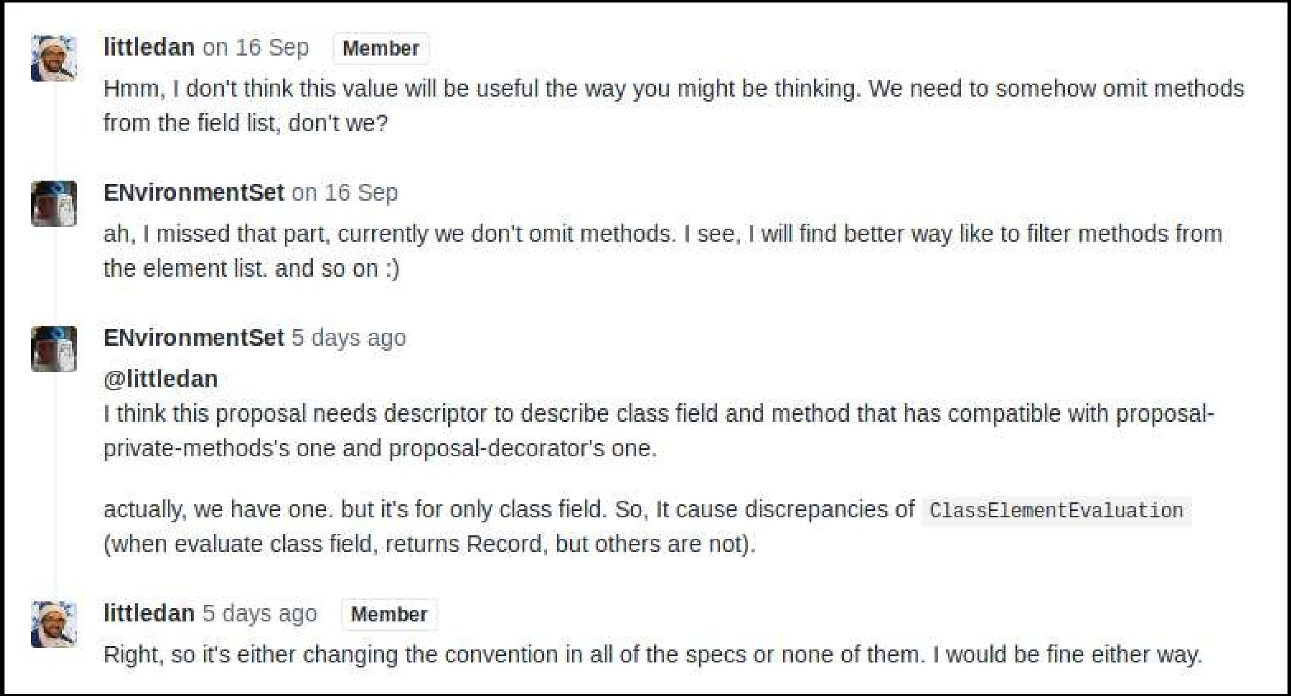
ClassElementEvaluation



```
  <emu-clause id="static-semantics-class-element-evaluation">
    <h1>Runtime Semantics: ClassElementEvaluation</h1>
    <p>With parameters _object_ and _enumerable_.</p>

    <emu-grammar>ClassElement : FieldDefinition `;`</emu-grammar>
    <emu-alg>
      1. Return ClassFieldDefinitionEvaluation of |FieldDefinition| with parameter _object_.
    </emu-alg>
    <emu-grammar>
        ClassElement : MethodDefinition
        ClassElement : `static` MethodDefinition
    </emu-grammar>
    <emu-alg>
-     1. Perform ? PropertyDefinitionEvaluation with parameters _object_ and _enumerable_.
-     1. Return an empty List.
+     1. Return PropertyDefinitionEvaluation with parameters _object_ and _enumerable_.
    </emu-alg>
  </emu-clause>
```

이후에 계속된 논의



littledan on 16 Sep Member

Hmm, I don't think this value will be useful the way you might be thinking. We need to somehow omit methods from the field list, don't we?

ENvironmentSet on 16 Sep

ah, I missed that part, currently we don't omit methods. I see, I will find better way like to filter methods from the element list. and so on :)

ENvironmentSet 5 days ago

@littledan
I think this proposal needs descriptor to describe class field and method that has compatible with proposal-private-methods's one and proposal-decorator's one.

actually, we have one. but it's for only class field. So, It cause discrepancies of `ClassElementEvaluation` (when evaluate class field, returns Record, but others are not).

littledan 5 days ago Member

Right, so it's either changing the convention in all of the specs or none of them. I would be fine either way.

END