functions

ECMAScript 2015

S67

일정 : 2016/9/22 ~ 16/10/27
시간 : 매주 목요일 저녁 8~10시
장소 : 네이버 D2 STARTUP FACTORY

projectBS x Bsidesoft x RoasteryKay x NAVER D2

Execution Context

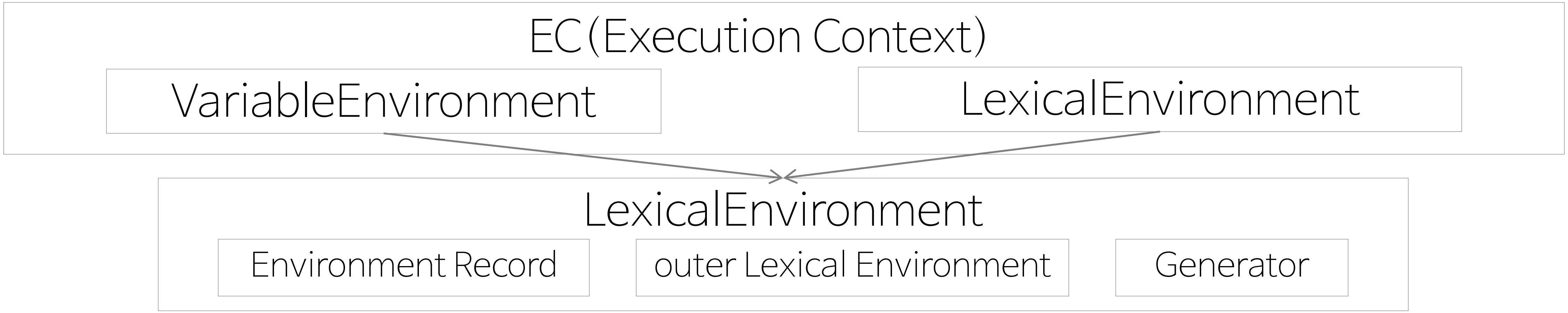
함수는 호출 시마다 고유한 실행컨텍스트를 생성한 뒤, 이를 활용하여 함수의 내용을 실행해간다.

```
const test = function() {};
test(); new EC_test()
```

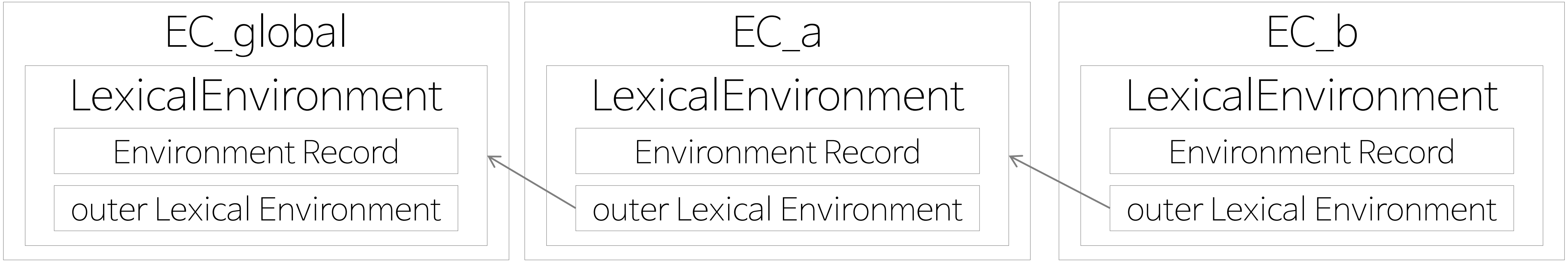
EC(Execution Context) = LexicalEnvironment
VariableEnvironment

code evaluation state
Function
Realm

LexicalEnvironment



```
const a = function() {
        const b = function() {}
        b();
};
a();
```



Environment Record Type

Object Environment Record

Declarative Environment Record

Global Environment Record

Function Environment Record

Module Environment Record

Environment Record

http://www.ecma-international.org/ecma-262/5.1/#sec-10.2.1
http://www.ecma-international.org/ecma-262/6.0/#sec-environment-records

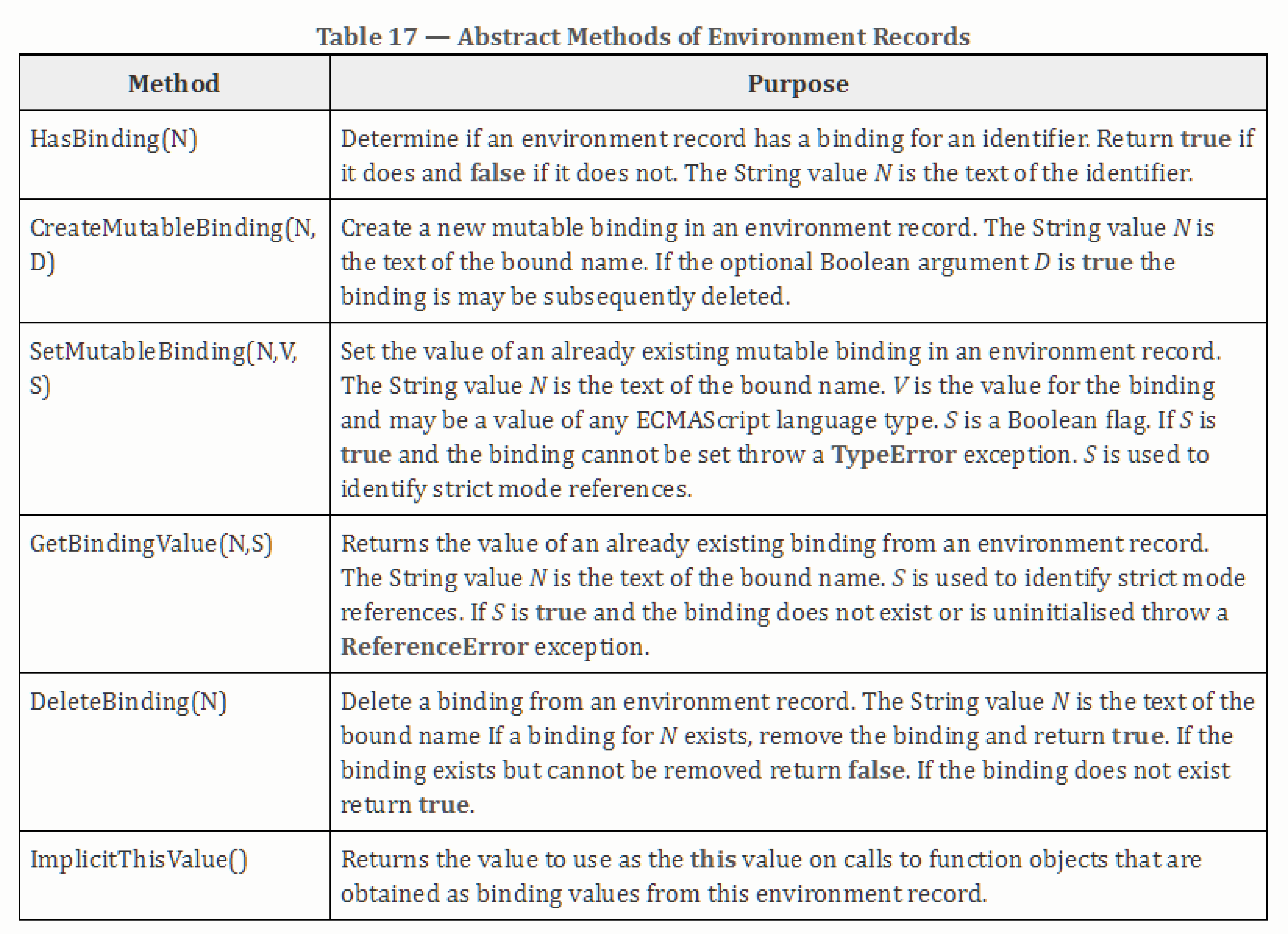


Table 17 — Abstract Methods of Environment Records

Method	Purpose
HasBinding(N)	Determine if an environment record has a binding for an identifier. Return **true** if it does and **false** if it does not. The String value *N* is the text of the identifier.
CreateMutableBinding(N, D)	Create a new mutable binding in an environment record. The String value *N* is the text of the bound name. If the optional Boolean argument *D* is **true** the binding is may be subsequently deleted.
SetMutableBinding(N,V, S)	Set the value of an already existing mutable binding in an environment record. The String value *N* is the text of the bound name. *V* is the value for the binding and may be a value of any ECMAScript language type. *S* is a Boolean flag. If *S* is **true** and the binding cannot be set throw a **TypeError** exception. *S* is used to identify strict mode references.
GetBindingValue(N,S)	Returns the value of an already existing binding from an environment record. The String value *N* is the text of the bound name. *S* is used to identify strict mode references. If *S* is **true** and the binding does not exist or is uninitialised throw a **ReferenceError** exception.
DeleteBinding(N)	Delete a binding from an environment record. The String value *N* is the text of the bound name If a binding for *N* exists, remove the binding and return **true**. If the binding exists but cannot be removed return **false**. If the binding does not exist return **true**.
ImplicitThisValue()	Returns the value to use as the **this** value on calls to function objects that are obtained as binding values from this environment record.

```
var foo = "abc";
  with({ foo: "bar" }) {
    function f() {
      console.log(foo);
    }
    f();
  }
```

ES5 V8 = 'abc'
ES5 firefox = 'bar'

Environment Record

http://www.ecma-international.org/ecma-262/5.1/#sec-10.2.1
http://www.ecma-international.org/ecma-262/6.0/#sec-environment-records

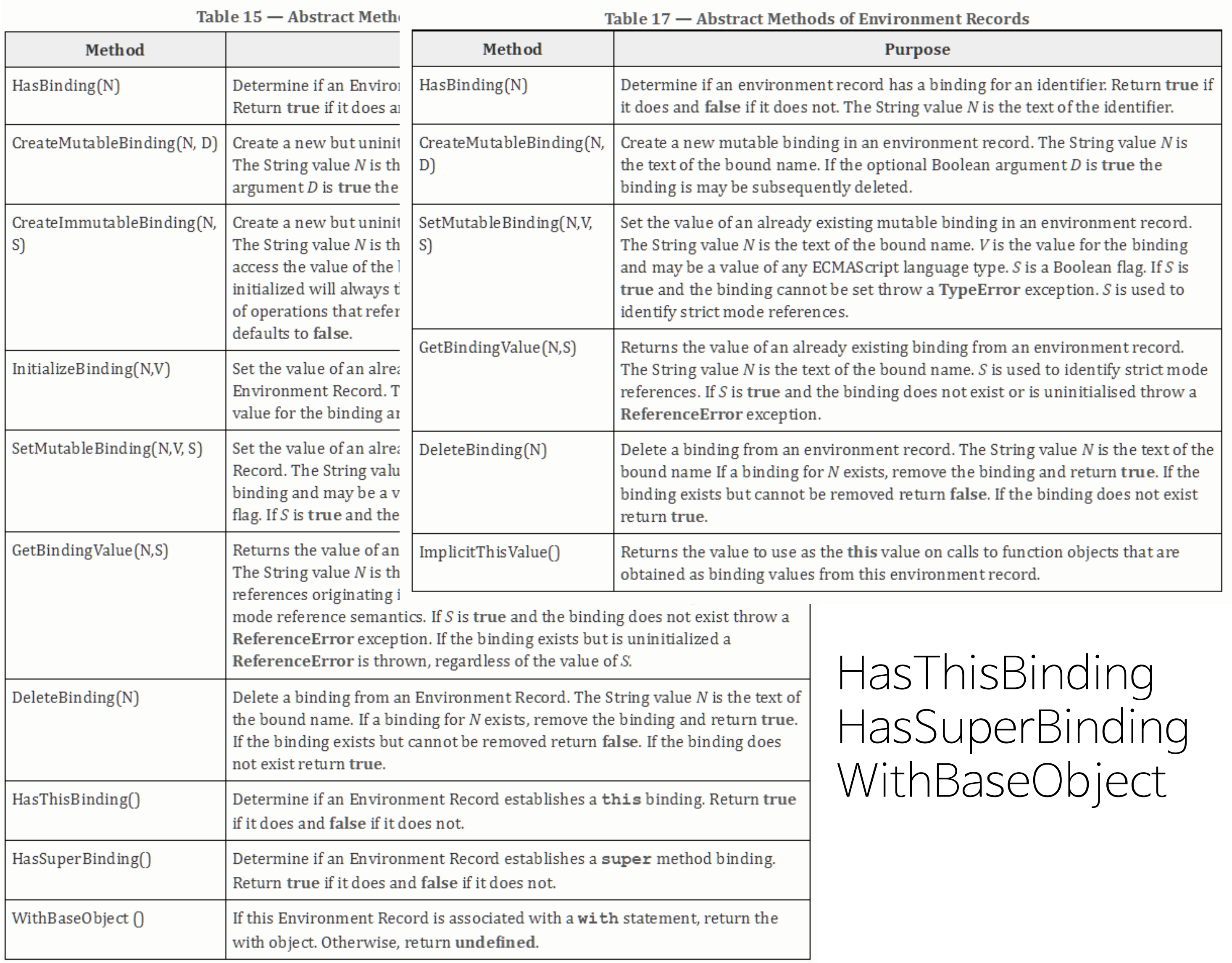


Table 15 — Abstract Meth

Method	
HasBinding(N)	Determine if an Enviro Return **true** if it does a
CreateMutableBinding(N, D)	Create a new but uninit The String value *N* is th argument *D* is **true** the
CreateImmutableBinding(N, S)	Create a new but uninit The String value *N* is th access the value of the l initialized will always t of operations that refer defaults to **false**.
InitializeBinding(N,V)	Set the value of an alre Environment Record. T value for the binding a
SetMutableBinding(N,V, S)	Set the value of an alre Record. The String valu binding and may be a v flag. If *S* is **true** and the
GetBindingValue(N,S)	Returns the value of an The String value *N* is th references originating i mode reference semantics. If *S* is **true** and the binding does not exist throw a **ReferenceError** exception. If the binding exists but is uninitialized a **ReferenceError** is thrown, regardless of the value of *S*.
DeleteBinding(N)	Delete a binding from an Environment Record. The String value *N* is the text of the bound name. If a binding for *N* exists, remove the binding and return **true**. If the binding exists but cannot be removed return **false**. If the binding does not exist return **true**.
HasThisBinding()	Determine if an Environment Record establishes a **`this`** binding. Return **true** if it does and **false** if it does not.
HasSuperBinding()	Determine if an Environment Record establishes a **`super`** method binding. Return **true** if it does and **false** if it does not.
WithBaseObject ()	If this Environment Record is associated with a **`with`** statement, return the with object. Otherwise, return **undefined**.

Table 17 — Abstract Methods of Environment Records

Method	Purpose
HasBinding(N)	Determine if an environment record has a binding for an identifier. Return **true** if it does and **false** if it does not. The String value *N* is the text of the identifier.
CreateMutableBinding(N, D)	Create a new mutable binding in an environment record. The String value *N* is the text of the bound name. If the optional Boolean argument *D* is **true** the binding is may be subsequently deleted.
SetMutableBinding(N,V, S)	Set the value of an already existing mutable binding in an environment record. The String value *N* is the text of the bound name. *V* is the value for the binding and may be a value of any ECMAScript language type. *S* is a Boolean flag. If *S* is **true** and the binding cannot be set throw a **TypeError** exception. *S* is used to identify strict mode references.
GetBindingValue(N,S)	Returns the value of an already existing binding from an environment record. The String value *N* is the text of the bound name. *S* is used to identify strict mode references. If *S* is **true** and the binding does not exist or is uninitialised throw a **ReferenceError** exception.
DeleteBinding(N)	Delete a binding from an environment record. The String value *N* is the text of the bound name If a binding for *N* exists, remove the binding and return **true**. If the binding exists but cannot be removed return **false**. If the binding does not exist return **true**.
ImplicitThisValue()	Returns the value to use as the **this** value on calls to function objects that are obtained as binding values from this environment record.

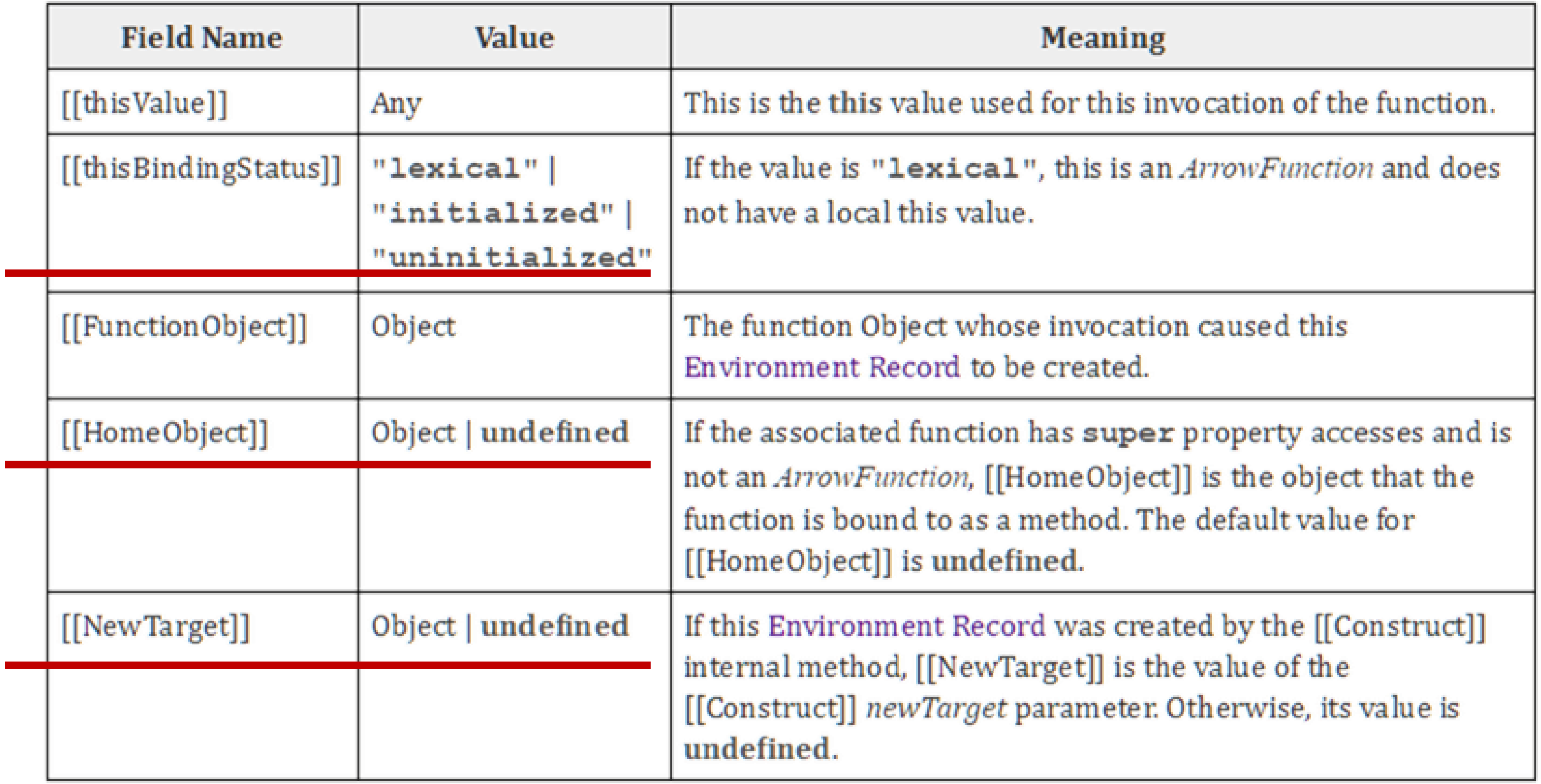
HasThisBinding
HasSuperBinding
WithBaseObject

```
var foo = "abc";
  with({ foo: "bar" }) {
    function f() {
      console.log(foo);
    }
    f();
  }
```

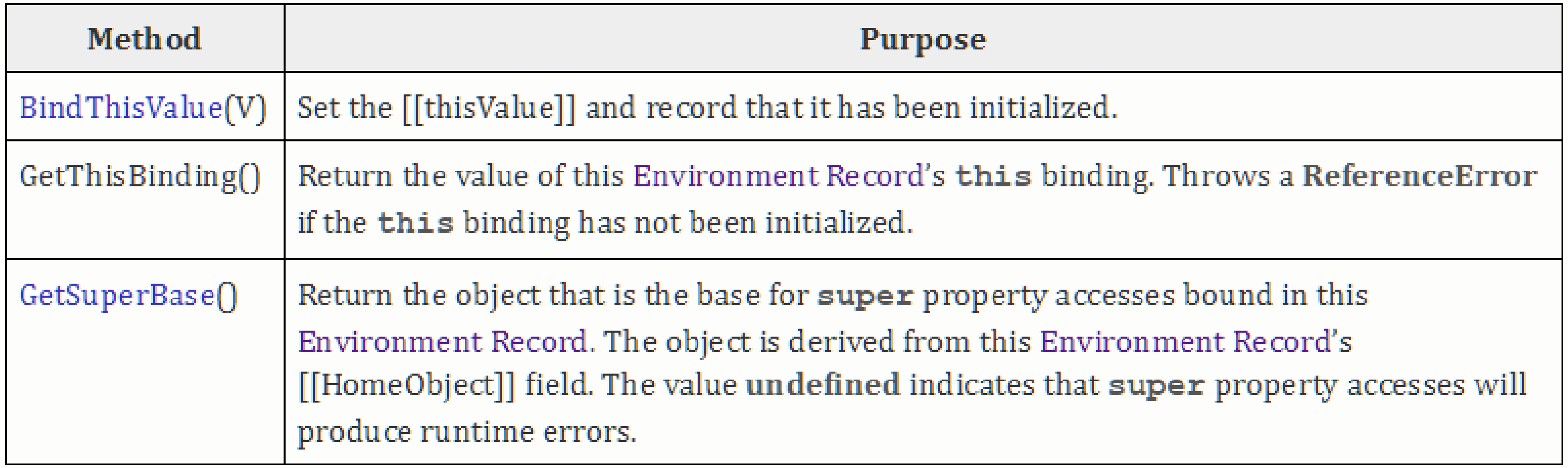
ES5 V8 = 'abc'
ES5 firefox = 'bar'

ES2015 = 'bar'

Function Environment Record



Field Name	Value	Meaning
[[thisValue]]	Any	This is the **this** value used for this invocation of the function.
[[thisBindingStatus]]	**`"lexical"`** | **`"initialized"`** | **`"uninitialized"`**	If the value is **`"lexical"`**, this is an *ArrowFunction* and does not have a local this value.
[[FunctionObject]]	Object	The function Object whose invocation caused this Environment Record to be created.
[[HomeObject]]	Object | **undefined**	If the associated function has **`super`** property accesses and is not an *ArrowFunction*, [[HomeObject]] is the object that the function is bound to as a method. The default value for [[HomeObject]] is **undefined**.
[[NewTarget]]	Object | **undefined**	If this Environment Record was created by the [[Construct]] internal method, [[NewTarget]] is the value of the [[Construct]] *newTarget* parameter. Otherwise, its value is **undefined**.



Method	Purpose
BindThisValue(V)	Set the [[thisValue]] and record that it has been initialized.
GetThisBinding()	Return the value of this Environment Record's **`this`** binding. Throws a **ReferenceError** if the **`this`** binding has not been initialized.
GetSuperBase()	Return the object that is the base for **`super`** property accesses bound in this Environment Record. The object is derived from this Environment Record's [[HomeObject]] field. The value **undefined** indicates that **`super`** property accesses will produce runtime errors.

arrow function

thisBinding을 동적으로 하지 않고 lexical Record의 것을 사용하도록 고정됨.

함수를 함수 순수 함수 호출 외에 사용할 수 없음

```
const a = 10;
const test =_=>this.a;

test.call({a:5});        동적 바인딩 금지

new test();              클래스는 class문으로 처리

const obj = {
  a:7,
  method:test            메소드 사용금지
};
obj.method();

el.addEventListener('click', e=>{
  //this                 리스너의 this 사용금지
});
```

Function type Role

class구문 - 클래스, 메소드를 생성함

arrow function - 람다를 생성함

object method - 해당 object의 메소드를 생성함

```
class{
  constructor() {}
  method() {}
}

const obj = {
  method() {}
}

const test =v=〉{}
```