



Google @fhinkel

8.1 OrdinaryGet (*O*, *P*, *Receiver*)

When the abstract operation OrdinaryGet is called with Object *O*, property key *P*, and an ECMAScript language value *Receiver*, the following steps are taken:

1. Assert: IsPropertyKey(*P*) is **true**.
2. Let *desc* be ? *O*.[[GetOwnProperty]](*P*).
3. If *desc* is **undefined**, then
 - a. Let *parent* be ? *O*.[[GetPrototypeOf]]().
 - b. If *parent* is **null**, return **undefined**.
 - c. Return ? *parent*.[[Get]](*P*, *Receiver*).
4. If IsDataDescriptor(*desc*) is **true**, return *desc*.[[Value]].
5. Assert: IsAccessorDescriptor(*desc*) is **true**.
6. Let *getter* be *desc*.[[Get]].
7. If *getter* is **undefined**, return **undefined**.
8. Return ? Call(*getter*, *Receiver*).

```
function add(obj) {  
    return 1 + obj.x;  
}
```

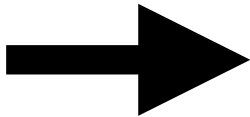
```
add({x: 42});
```

```
add({x: 7});
```

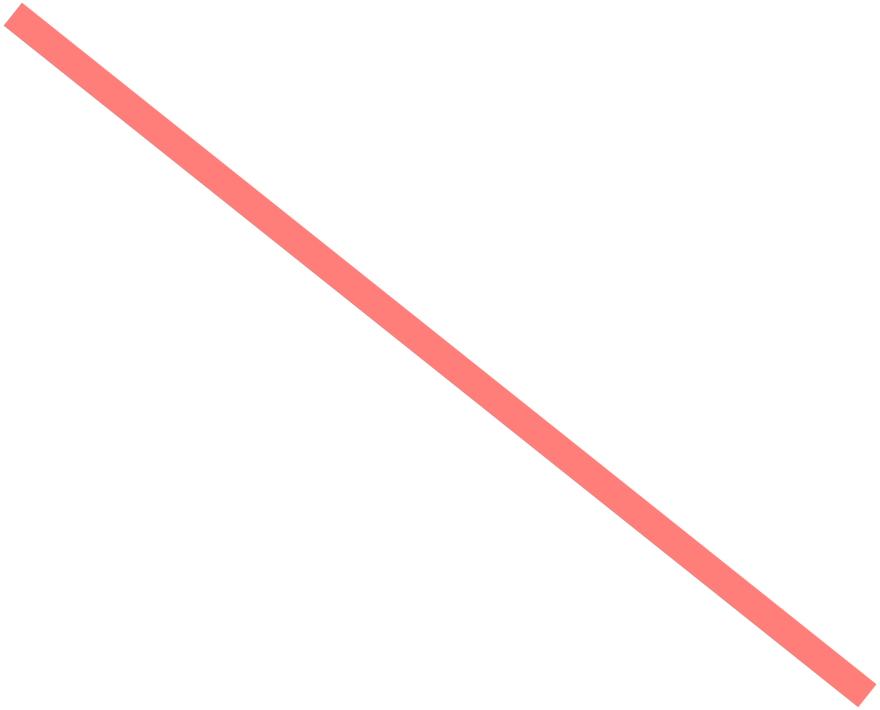


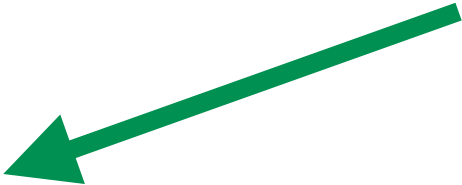

Cache of shortcuts

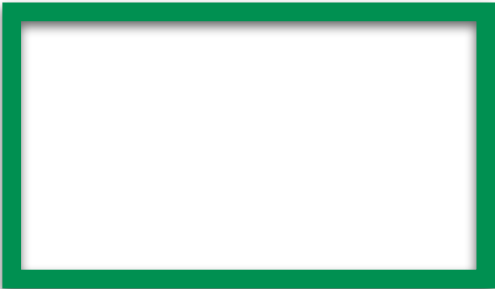
{x: int}



offset 1











AST

```
graph TD; AST[AST] --> BC[Baseline Compiler]; BC --> MC[Machine code]; OC[Optimizing Compiler] --> OMC[Optimized Machine Code];
```

The diagram illustrates the compilation process. It starts with an Abstract Syntax Tree (AST) at the top left. An arrow points from the AST to a rounded rectangle labeled 'Baseline Compiler'. From the 'Baseline Compiler', an arrow points down to a rectangle labeled 'Machine code'. To the right of the 'Baseline Compiler' is another rounded rectangle labeled 'Optimizing Compiler'. An arrow points from the 'Optimizing Compiler' down to a rectangle labeled 'Optimized Machine Code'.

Baseline
Compiler

Machine code

Optimizing
Compiler

Optimized
Machine Code

Cache of
shortcuts

```
function add(obj) {  
  return 1 + obj.x;  
}
```

{x: int}

offset 1

```
add({x: 42});
```

```
add({x: 7});
```

8.1 OrdinaryGet (*O*, *P*, *Receiver*)

When the abstract operation OrdinaryGet is called with Object *O*, property key *P*, and an ECMAScript language value *Receiver*, the following steps are taken:

1. Assert: IsPropertyKey(*P*) is true.
2. Let *desc* be ? *O*.[[GetOwnProperty]](*P*).
3. If *desc* is undefined, then
 - a. Let *parent* be ? *O*.[[GetPrototypeOf]]().
 - b. If *parent* is null, return undefined.
 - c. Return ? *parent*.[[Get]](*P*, *Receiver*).
4. If IsDataDescriptor(*desc*) is true, return *desc*.[[Value]].
5. Assert: IsAccessorDescriptor(*desc*) is true.
6. Let *getter* be *desc*.[[Get]].
7. If *getter* is undefined, return undefined.
8. Return ? Call(*getter*, *Receiver*).