12.11 The `switch` Statement

Syntax

SwitchStatement **:**
 `switch (` *Expression* **`)`** *CaseBlock*

CaseBlock **:**
 `{` *CaseClauses*$_{opt}$ **`}`**
 `{` *CaseClauses*$_{opt}$ *DefaultClause* *CaseClauses*$_{opt}$ **`}`**

CaseClauses **:**
 CaseClause
 CaseClauses CaseClause

CaseClause **:**
 `case` *Expression* **`:`** *StatementList*$_{opt}$

DefaultClause **:**
 `default :` *StatementList*$_{opt}$

Semantics

The production *SwitchStatement* **:** **`switch (`** *Expression* **`)`** *CaseBlock* is evaluated as follows:

1. Evaluate *Expression*.
2. Call GetValue(Result(1)).
3. Evaluate *CaseBlock*, passing it Result(2) as a parameter.
4. If Result(3).type is **break** and Result(3).target is in the current label set, return (**normal**, Result(3).value, **empty**).
5. Return Result(3).

The production *CaseBlock* **: `{`** *CaseClauses*$_{opt}$ **`}`** is given an input parameter, *input*, and is evaluated as follows:

1. Let *V* = **empty**.
2. Let *A* be the list of *CaseClause* items in source text order.
3. Let *C* be the next *CaseClause* in *A*. If there is no such *CaseClause*, then go to step 16.
4. Evaluate *C*.
5. If *input* is not equal to Result(4) as defined by the **`!==`** operator, then go to step 3.
6. If *C* does not have a *StatementList*, then go to step 10.
7. Evaluate *C*'s *StatementList* and let *R* be the result.
8. If *R* is an abrupt completion, then return *R*.
9. Let *V* = *R*.value.
10. Let *C* be the next *CaseClause* in *A*. If there is no such *CaseClause*, then go to step 16.
11. If *C* does not have a *StatementList*, then go to step 10.
12. Evaluate *C*'s *StatementList* and let *R* be the result.
13. If *R*.value is not **empty**, then let *V* = *R*.value.
14. If *R* is an abrupt completion, then return (*R*.type, *V*, *R*.target).
15. Go to step 10.
16. Return (**normal**, *V*, **empty**).

The production *CaseBlock* **: `{`** *CaseClauses*$_{opt}$ *DefaultClause CaseClauses*$_{opt}$ **`}`** is given an input parameter, *input*, and is evaluated as follows:

1. Let *V* = **empty**.

2. Let *A* be the list of *CaseClause* items in the first *CaseClauses*, in source text order.
3. Let *C* be the next *CaseClause* in *A*. If there is no such *CaseClause*, then go to step 11.
4. Evaluate *C*.
5. If *input* is not equal to Result(4) as defined by the **!==** operator, then go to step 3.
6. If *C* does not have a *StatementList*, then go to step 20.
7. Evaluate *C*'s *StatementList* and let *R* be the result.
8. If *R* is an abrupt completion, then return *R*.
9. Let *V* = *R*.value.
10. Go to step 20.
11. Let *B* be the list of *CaseClause* items in the second *CaseClauses*, in source text order.
12. Let *C* be the next *CaseClause* in *B*. If there is no such *CaseClause*, then go to step 26.
13. Evaluate *C*.
14. If *input* is not equal to Result(13) as defined by the **!==** operator, then go to step 12.
15. If *C* does not have a *StatementList*, then go to step 31.
16. Evaluate *C*'s *StatementList* and let *R* be the result.
17. If *R* is an abrupt completion, then return *R*.
18. Let *V* = *R*.value.
19. Go to step 31.
20. Let *C* be the next *CaseClause* in *A*. If there is no such *CaseClause*, then go to step 26.
21. If *C* does not have a *StatementList*, then go to step 20.
22. Evaluate *C*'s *StatementList* and let *R* be the result.
23. If *R*.value is not **empty**, then let *V* = *R*.value.
24. If *R* is an abrupt completion, then return (*R*.type, *V*, *R*.target).
25. Go to step 20.
26. If the *DefaultClause* does not have a *StatementList*, then go to step 30.
27. Evaluate the *DefaultClause*'s *StatementList* and let *R* be the result.
28. If *R*.value is not **empty**, then let *V* = *R*.value.
29. If *R* is an abrupt completion, then return (*R*.type, *V*, *R*.target).
30. Let *B* be the list of *CaseClause* items in the second *CaseClauses*, in source text order.
31. Let *C* be the next *CaseClause* in *B*. If there is no such *CaseClause*, then go to step 37.
32. If *C* does not have a *StatementList*, then go to step 31.
33. Evaluate *C*'s *StatementList* and let *R* be the result.
34. If *R*.value is not **empty**, then let *V* = *R*.value.
35. If *R* is an abrupt completion, then return (*R*.type, *V*, *R*.target).
36. Go to step 31.
37. Return (**normal**, *V*, **empty**).

The production *CaseClause* **:** **case** *Expression* **:** *StatementList*$_{opt}$ is evaluated as follows:

1. Evaluate *Expression*.
2. Call GetValue(Result(1)).
3. Return Result(2).

NOTE Evaluating *CaseClause* does not execute the associated *StatementList*. It simply evaluates the *Expression* and returns the value, which the *CaseBlock* algorithm uses to determine which *StatementList* to start executing.