```
switch (op) {
    case Operator.ADD:
        // execute add
        break;
    case Operator.DIV:
        // execute div
        break;
    // ...
}
```

For this switch statement, the compiler will generate the following code.

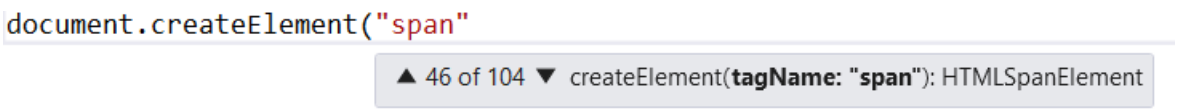
```
switch (op) {
    case 0 /* Operator.ADD */ :
        // execute add
        break;
    case 1 /* Operator.DIV */ :
        // execute div
        break;
    // ...
}
```

JavaScript implementations can use these explicit constants to generate efficient code for this switch statement, for example by building a jump table indexed by case value.

1.8 Overloading on String Parameters

An important goal of TypeScript is to provide accurate and straightforward types for existing JavaScript programming patterns. To that end, TypeScript includes generic types, discussed in the next section, and *overloading on string parameters*, the topic of this section.

JavaScript programming interfaces often include functions whose behavior is discriminated by a string constant passed to the function. The Document Object Model makes heavy use of this pattern. For example, the following screen shot shows that the 'createElement' method of the 'document' object has multiple signatures, some of which identify the types returned when specific strings are passed into the method.



The following code fragment uses this feature. Because the 'span' variable is inferred to have the type 'HTMLSpanElement', the code can reference without static error the 'isMultiline' property of 'span'.

```
var span = document.createElement("span");
span.isMultiLine = false;  // OK: HTMLSpanElement has an 'isMultiline' property
```