Expressions in a JavaScript Switch Case

Posted At : December 29, 2015 1:29 PM | Posted By : Jon Hartmann
Related Categories: Experiments, Javascript

Today I stumbled across something I'd never seen before, and at first thought had to be an error - a switch statement with expressions embedded in the case statement:

Edit in JSFiddle

- JavaScript

```
var percent = 33;
var answer;

switch (true){
  case (percent > 66.66 && percent <= 100):
    answer = '1';
    break;
  case (percent > 33.33 && percent <= 66.66):
    answer = '2';
    break;
  case (percent > 0 && percent <= 33.33):
    answer = '3';
    break;
}

console.log('Answer' + answer);
```

Wh
the heck is that thing?

Lets take a look at a normal example of a Switch statement:

Edit in JSFiddle

- JavaScript

```
var value = 'duck';
var answer;

switch (value){
  case 'camel':
    answer = '1';
    break;
  case 'dog':
    answer = '2';
    break;
  case 'duck':
    answer = '3';
    break;
}

console.log('Answer' + answer);
```

Compared to a normal Caste statement, this thing is pretty weird. So weird that my first instinct was that this code was wrong and would break, but it was located in a very active part of the codebase - there was no way that it was failing to do what it it was trying to do... but how was it resolving the right value?

Normally I think of a Switch as a kind of stand in for a key-value lookup. Ideally, the run time is smart enough to evaluate it like that rather than checking each value sequentially (I'll let some other blog tell you which browsers do it best), but there is no way to actually do that optimization if the value of the Case is an expression - at best you would have to calculate the answers to each case until you found one that matched... and since a Switch matches Cases sequentially, you could in theory stop with the first match.

I created a test to see if that was true:

Edit in JSFiddle

- JavaScript

```
var percent = 33;
var answer;

function range1(percent) {
 console.log('Evaluate 1');
 return (percent > 66.66 && percent <= 100);
}

function range2(percent) {
 console.log('Evaluate 2');
 return (percent > 33.33 && percent <= 66.66);
}

function range3(percent) {
 console.log('Evaluate 3');
```

As you can see from the result, my expectation was correct - the browser has to evaluate each statement individually and sequentially in order to find a match... which makes this the exact same execution as a simple If-Else block:

Edit in JSFiddle

- JavaScript

```
var percent = 55;
var answer;

function range1(percent) {
 console.log('Evaluate 1');
 return (percent > 66.66 && percent <= 100);
}

function range2(percent) {
 console.log('Evaluate 2');
 return (percent > 33.33 && percent <= 66.66);
}

function range3(percent) {
 console.log('Evaluate 3');
 return (percent > 0 && percent <= 33.33);
```

Which in the end means that this kind of Expression-in-Case setup is no better than the If-Else block. With nothing to show as an upside, applying the KISS methodology means we're better off with this less clever version:

Edit in JSFiddle

- JavaScript

```
var percent = 55;
var answer;

if (percent > 66.66 && percent <= 100) {
  answer = 'Answer 1';
} else if (percent > 33.33 && percent <= 66.66) {
  answer = 'Answer 2';
} else if (percent > 0 && percent <= 33.33) {
  answer = 'Answer 3';
}

console.log('Answer', answer)
```