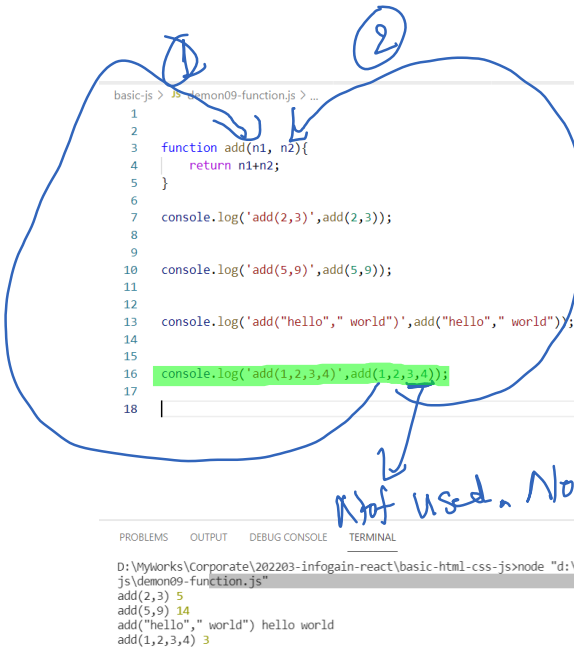
Javascript Functions

Thursday, March 10, 2022 2:36 PM



Parameters don't have well-defined data type

- We may pass any 'typeof' value and it will still work
- The result may not always be what you expect but it still works

Formal Parameters and Actual argument need not match in number

- You can pass more or less parameters than what a function expects.
- If you pass more parameters, the additional parameters will be *ignored*
  - *Ignored is not exactly correct*
- They **may not** be used by the function, but will not cause any error
- In this example, first two arguments will be assigned to n1, and n2.
  - Remaining arguments will not be used.
- If you pass fewer arguments, the remaining argument will be considered 'undefined'

```
var  x=  Add(5);   //same as calling   Add(5, undefined);
```

Additionally supplied Arguments are not really Lost

- The additional parameters that you pass to a function is NOT really lost.
- They are not accessible by formal parameters, because there is no formal parameter
- Javascript passes a hidden parameter to every function called **arguments**
- **arguments** is an array like object
  - It is not really an array
  - **It doesn't work with for-of loop**
  - **It can work with regular for loop**
- **Arguments** shall contain all arguments supplied
  - event those that are assigned to formal parameters

```
function  add( x, y )
{
    var arguments=[ /* list of all arguments */ ];

    return x+y;

}
```

```
add(2, 9);  // x--> 2,  y--> 9,  arguments: [2,9]

add(2, 8, 4, 1, 7);  // x--> 2,    y–>8,  arguments: [2,8,4,1,7 ]
```

Note

- Arguments is NOT really an array
- It is an array like Object
- Actually it will look like
  - `{ 0: 2,  1 : 9 }`

What can be the use case of 'arguments'?

- We can write functions that may take any number of arguments

- Example
  - Write an Average function that can average all the numbers passed to it
    - average(2,3)
    - average(2,3,9,8,11,4,2)