	Bước	call stack	macrotask queue	microtask queue	web APIs
Sync	1	clearOutput()			
	2	output("BEGIN sync")			
	3	sFunction();			
	4	output(ret="Hello S!");			
	5	output("END.")			

	Bước	call stack	macrotask queue	microtask queue	web APIs
Timeout	1	clearOutput()			
	2	output("BEGIN settimeout");			
	3	setTimeout(function() { let ret = sFunction(); output(ret); }, 0);			
	4				function()
	5		sFunction();		
			output(ret);		
	6	output("END.");			
	7	sFunction();	output(ret);		
	8	output(ret="Hello S!");			

	Bước	call stack	macrotask queue	microtask queue
Promise	1	clearOutput()		
	2	output("BEGIN promise")		
	3	aFunction1().then(function (ret) { output(ret); });		
	4	output("END.")	output(ret);	new Promise(function(fulfill, reject) { fulfill("Hello A1!"); })
	5	fulfill("Hello A1!"); //ret="Hello A1!"	output(ret);	
	6	output(ret);		

	Bước	call stack	macrotask queue	microtask queue
Async	1	clearOutput()		
	2	output("BEGIN async")		
	3	aFunction2().then(function (ret) { output(ret); });		
	4	output("END.")	output(ret);	new Promise(function(fulfill, reject) { fulfill("Hello A2!"); })
	5	fulfill("Hello A2!"); //ret="Hello A2!"	output(ret);	
	6	output(ret);		

	Bước	call stack	macrotask queue	microtask queue
Await	1	clearOutput()		
	2	output("BEGIN await")		
	3	aFunction1();		
	4	//chờ aFunction() trả về xong kết quả		new Promise(function(fulfill, reject) { fulfill("Hello A1!"); })
	5	fulfill("Hello A1!"); //ret="Hello A1!"		
	6	output(ret);		
	7	output("END.");		

	Bước	call stack	macrotask queue	microtask queue
Async vs sync	1	clearOutput()		
	2	output("BEGIN sync vs async")		
	3	aFunction1().then(function (ret) { output(ret); });		
	4	aFunction2().then(function (ret) { output(ret); });	output(ret);	new Promise(function(fulfill, reject) { fulfill("Hello A1!"); })
	5		output(ret);	new Promise(function(fulfill, reject) { fulfill("Hello A1!"); })
			output(ret);	new Promise(function(fulfill, reject) { fulfill("Hello A2!"); })
	6	sFunction();	output(ret);	new Promise(function(fulfill, reject) { fulfill("Hello A1!"); })
			output(ret);	new Promise(function(fulfill, reject) { fulfill("Hello A2!"); })
	7	output(ret="Hello S!");	output(ret);	new Promise(function(fulfill, reject) { fulfill("Hello A1!"); })
			output(ret);	new Promise(function(fulfill, reject) { fulfill("Hello A2!"); })
	8	output("END.")	output(ret);	new Promise(function(fulfill, reject) { fulfill("Hello A1!"); })
			output(ret);	new Promise(function(fulfill, reject) { fulfill("Hello A2!"); })
	9	fulfill("Hello A1!"); //ret="Hello A1!"	output(ret);	new Promise(function(fulfill, reject) { fulfill("Hello A2!"); })
			output(ret);	
	10	output(ret);	output(ret);	new Promise(function(fulfill, reject) { fulfill("Hello A2!"); })
	11	fulfill("Hello A2!"); //ret="Hello A2!"	output(ret);	
	12	output(ret);		

	Bước	call stack	macrotask queue	microtask queue
Wait all	1	clearOutput()		
	2	output("BEGIN waitall");		
	3	aFunction1();		
	4	aFunction2();		new Promise(function(fulfill, reject) { fulfill("Hello A1!"); })
	5	Promise.all([p1, p2]).then(function (arr) { output(arr[0]); output(arr[1]); });		new Promise(function(fulfill, reject) { fulfill("Hello A1!"); })
				new Promise(function(fulfill, reject) { fulfill("Hello A2!"); })
	6	output("END.");	output(arr[0]); output(arr[1]);	new Promise(function(fulfill, reject) { fulfill("Hello A1!"); })
				new Promise(function(fulfill, reject) { fulfill("Hello A2!"); })
				Promise.all([p1, p2])
	7	fulfill("Hello A1!");	output(arr[0]); output(arr[1]);	new Promise(function(fulfill, reject) { fulfill("Hello A2!"); })
				Promise.all([p1, p2])
	8	fulfill("Hello A2!"); //arr=["Hello A1!", "Hello A2!"];	output(arr[0]); output(arr[1]);	Promise.all([p1, p2])
		//Promise.all() hoàn thành do cả p1 và p2 đã thành công		
	9	output(arr[0]); output(arr[1]);		

	Bước	call stack	macrotask queue	microtask queue	web APIs
Mixed	1	clearOutput()			
	2	output("BEGIN mixed");			
	3	setTimeout(() => { output('setTimeout 1'); }, 10);			
	4	setTimeout(() => { output('setTimeout 2'); }, 0);			() => { output('setTimeout 1'); }, 10
	5		output(res); return "Sub Promise 1";}	new Promise() //promise 1	() => { output('setTimeout 1'); }, 10
					() => { output('setTimeout 2'); }, 0
	6		output(res); return "Sub Promise 1";}	new Promise() //promise 1	() => { output('setTimeout 1'); }, 10
			output(res); return "Sub Promise 2";}	new Promise() //promise 2	() => { output('setTimeout 2'); }, 0
	7	fulfill('Promise 1');	output(res); //ret="Promise 1"; return "Sub Promise 1";}	new Promise() //promise 2	() => { output('setTimeout 1'); }, 10
			output(res); return "Sub Promise 2";}		() => { output('setTimeout 2'); }, 0
	8	fulfill('Promise 2');	output(res); //ret="Promise 1"; return "Sub Promise 1";}		() => { output('setTimeout 1'); }, 10
			output(res); //ret="Promise 2"; return "Sub Promise 2";}		() => { output('setTimeout 2'); }, 0
	9	output(res="Promise 1");	return "Sub Promise 1";	new Promise() //sub promise 1	() => { output('setTimeout 1'); }, 10
			output(res); //ret="Promise 2"; return "Sub Promise 2";}		() => { output('setTimeout 2'); }, 0
			output(res);		
	10	output(res="Promise 2");	return "Sub Promise 2";	new Promise() //sub promise 1	() => { output('setTimeout 1'); }, 10
			output(res);	new Promise() //sub promise 2	() => { output('setTimeout 2'); }, 0
			output(res);		
	11	fulfill('Sub Promise 1');	output(res); //ret="Sub Promise 1"	new Promise() //sub promise 2	() => { output('setTimeout 1'); }, 10
			output(res);		() => { output('setTimeout 2'); }, 0
	12	fulfill('Sub Promise 2');	output(res); //ret="Sub Promise 1"		() => { output('setTimeout 1'); }, 10
			output(res); //ret="Sub Promise 2"		() => { output('setTimeout 2'); }, 0
	13	output(res="Sub Promise 1");	output(res); //ret="Sub Promise 2"		() => { output('setTimeout 1'); }, 10
					() => { output('setTimeout 2'); }, 0
	14	output(res="Sub Promise 2");			() => { output('setTimeout 1'); }, 10
					() => { output('setTimeout 2'); }, 0
	15		output('setTimeout 2');		() => { output('setTimeout 1'); }, 10
	16	output('setTimeout 2');	output('setTimeout 1');		
	17	output('setTimeout 1');			