3. 异步 JavaScript

并发 ≠ 异步编程

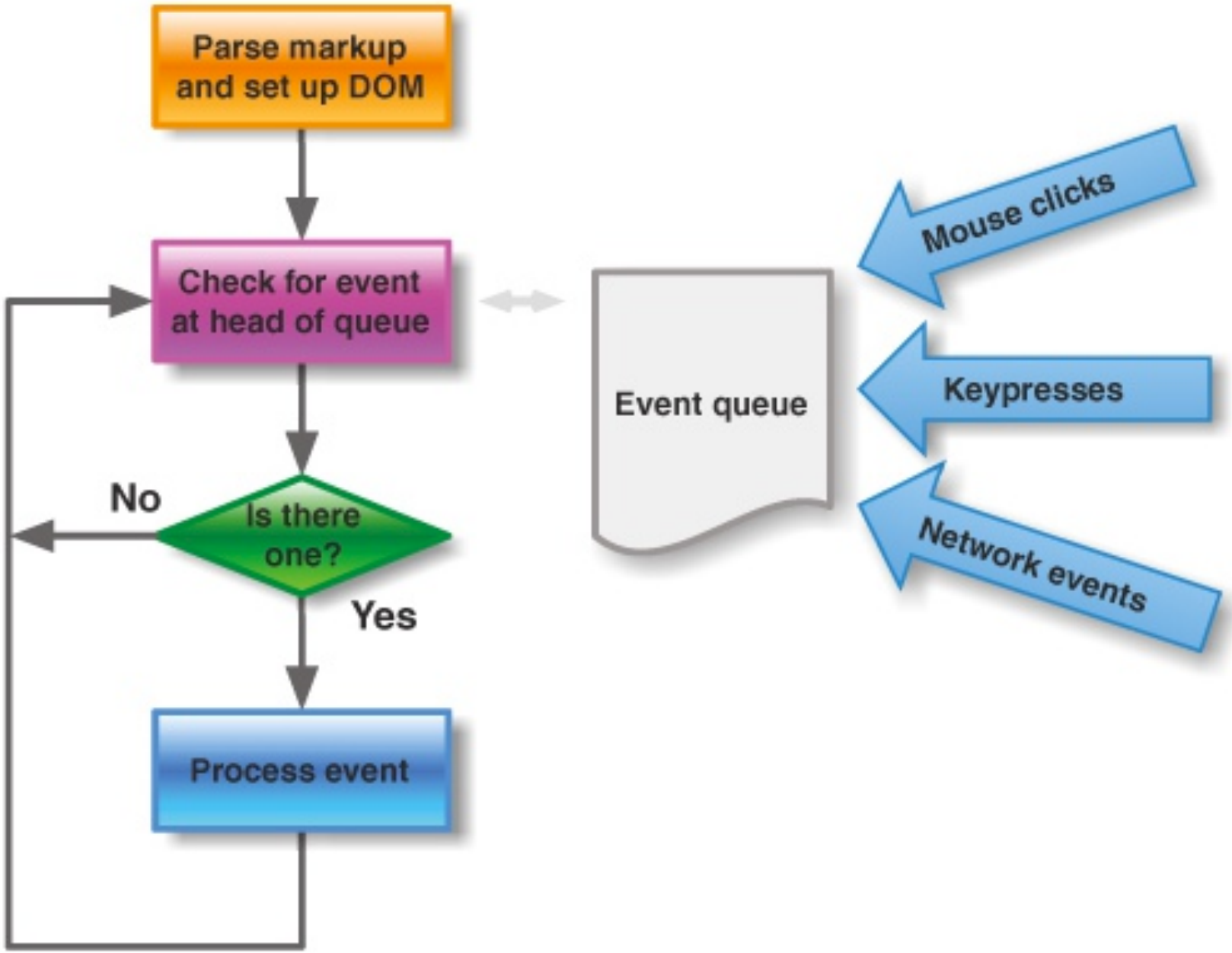
- 并发指两个以上的计算过程在资源共享的情况下同时执行的能力。
- 这些处理过程（有时也叫线程）或者共享一个处理器，或者分布在网络中由其他执行管理器异步执行。
- 并行进程间的通信一般是显示的，要么通过消息传递，要么通过共享变量。

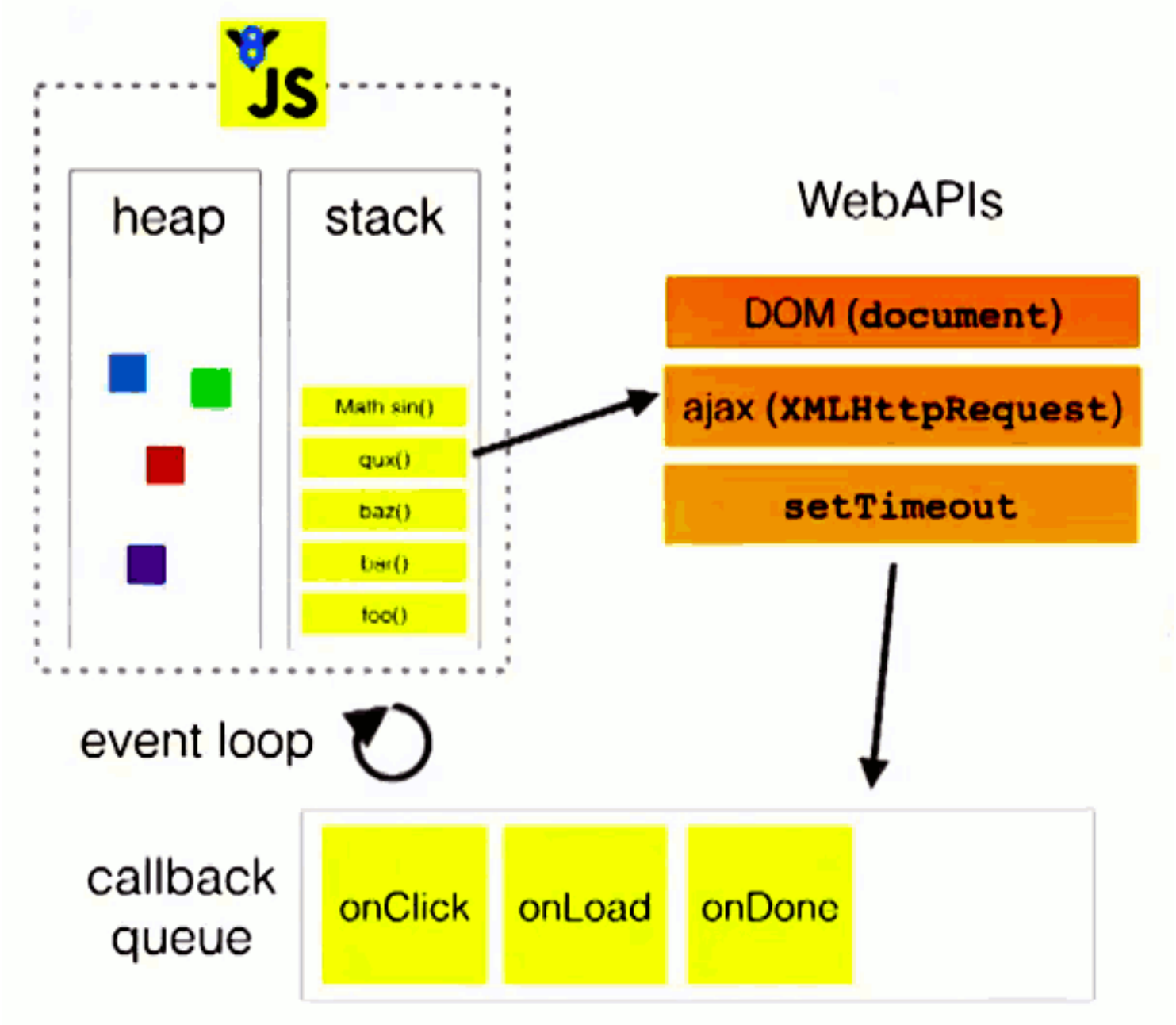
单线程的语言

- 简单（不用担心竞争和死锁）
- 脚本无法利用全部内核
- 脚本必须定期让位给浏览器的UI界面

事件循环

- 将单线程的能力发挥到极限
- HTML5提出Web Worker标准，允许JavaScript脚本创建多个线程，但是子线程完全受主线程控制，且不得操作DOM。所以，这个新标准并没有改变JavaScript单线程的本质。


Parse markup and set up DOM
Check for event at head of queue
Is there one?
No
Yes
Process event
Event queue
Mouse clicks
Keypresses
Network events


JS
heap
stack
Math.sin()
qux()
baz()
ber()
foo()
WebAPIs
DOM (document)
ajax (XMLHttpRequest)
setTimeout
event loop
callback queue
onClick
onLoad
onDone

名词解释

- 堆：堆是内存中的顺序无关的容器。堆是 JavaScript 存放正在使用或未被垃圾回收清理的变量和对象的地方。
- 帧：帧是事件循环周期中需要被连续执行的工作单元。帧包含把函数对象和堆中的变量链接在一起的执行上下文。
- 栈：事件循环栈包含了执行一个消息所需的所有连续的帧。事件循环自顶向下处理帧。
- 队列：队列是等待处理的消息的列表。

回调的用途

- For asynchronous execution (such as reading files, and making HTTP requests)
- In Event Listeners/Handlers
- In setTimeout and setInterval methods
- For Generalization: code conciseness

For asynchronous execution

```
var req = new XMLHttpRequest();
req.open('GET', url);
req.onload = function (){};
req.onerror = function (){};
req.send();

// 另一种写法
var req = new XMLHttpRequest();
req.open('GET', url);
req.send();
req.onload = function (){};
req.onerror = function (){};
```

In Event Listeners/Handlers

```
// jQuery
$("#btn_1").click(function() {
  alert("Btn 1 Clicked");
});

// Or
$(“#btn_1”).on(“click”, onClickHandler);
function onClickHandler(){
…
}
```

In setTimeout and setInterval methods

```
console.log(1);
setTimeout(function(){console.log(2);},1000);
console.log(3);
```

For Generalization: code conciseness

```
function successCallback() {
    // Do stuff before send
}

function successCallback() {
    // Do stuff if success message received
}

function completeCallback() {
    // Do stuff upon completion
}

function errorCallback() {
    // Do stuff if error received
}

$.ajax({
    url:"http://fiddle.jshell.net/favicon.png",
    success:successCallback,
    complete:completeCallback,
    error:errorCallback

});
```

回调地狱

```
var p_client = new Db('integration_tests_20', new Server("127.0.0.1", 27017, {}),
{'pk':CustomPKFactory});
p_client.open(function(err, p_client) {
    p_client.dropDatabase(function(err, done) {
        p_client.createCollection('test_custom_key', function(err, collection) {
            collection.insert({'a':1}, function(err, docs) {
                collection.find({'_id':new ObjectID("aaaaaaaaaaaa")}, function(err, cursor) {
                    cursor.toArray(function(err, items) {
                        test.assertEquals(1, items.length);

                        // Let's close the db
                        p_client.close();
                    });
                });
            });
        });
    });
});
```