Asynchronous JavaScript

@子回 <webmaster@leapoahead.com>

吴迪（子回）

前端开发者

- 同济大学软件学院
- 社区爱好者
- 热爱结交小伙伴
- WeChat: leapoahead

Lectures

- Basics
- Patterns
- Reactive Programming

Got question?

- Raise your hand, or
- Leave it at http://wudi.link/baF60b

ES6 Primer

- Arrow function (http://wudi.link/67a368)

- Not just a syntax sugar

Asynchronous JavaScript Basics

Lecture 1

```
var startTime = Date.now();
setTimeout(function() {
    var endTime = Date.now();
    console.log(endTime - startTime);
}, 1000);
```

```
var startTime = Date.now();
setTimeout(function() {
    var endTime = Date.now();
    console.log(endTime - startTime);
}, 1000);
for (var i = 0; i < 1000000000; i ++) {}
```

JavaScript is …

- Single-threaded! **(fundamentally)**
- Should be **non-blocking** **(hold on …)**

As example

Chrome is ... Multi-threaded!

- ui_thread, io_thread
- file_thread, db_thread
- safe_browsing_thread
- ... more threads

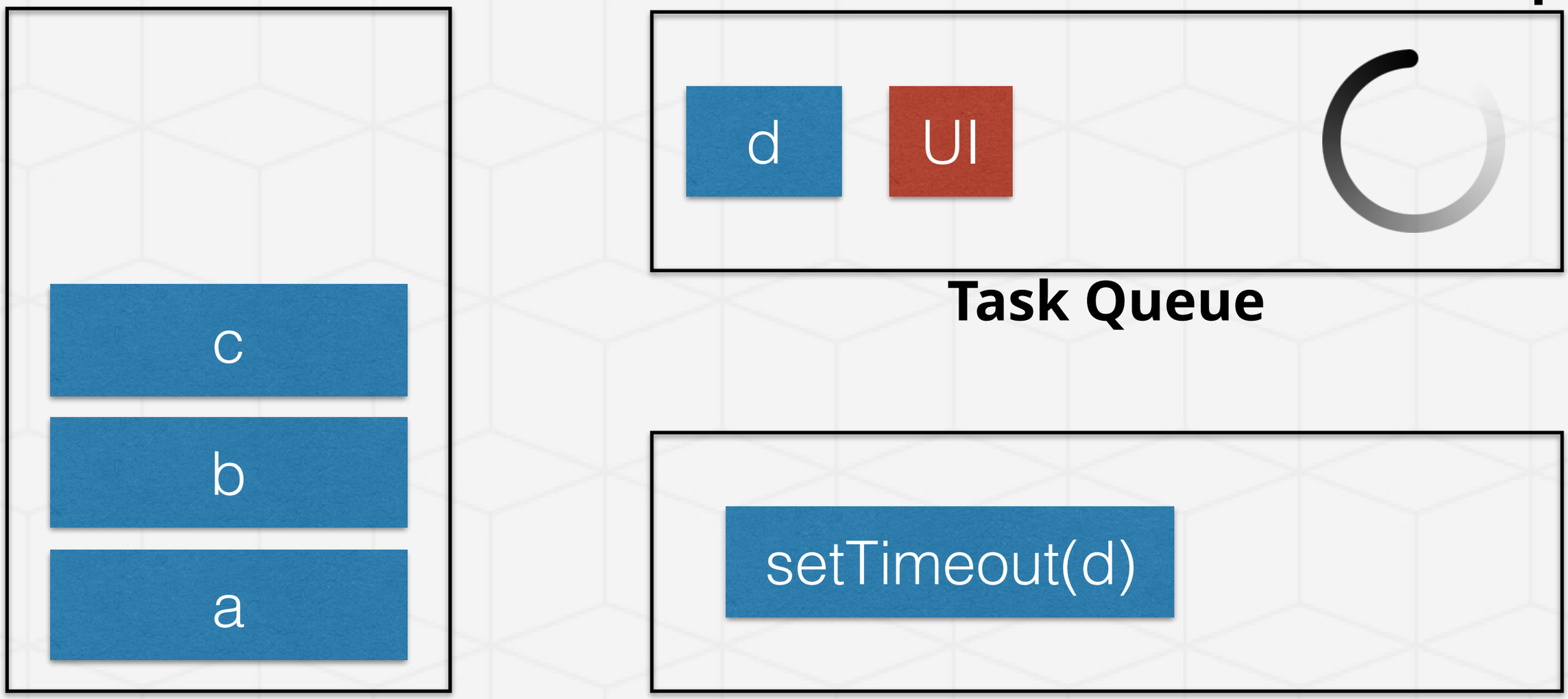
JS runs in main thread!

> Style, layout, and some paint setup operations run on the renderer's main thread, which is the same place that JavaScript runs.
>
> — Chrome Developer Documentations

```
//app.use(favicon(__dirname + '/public/favicon.ico'));
app.use(logger('dev'));
app.use(bodyParser.json());
app.use(bodyParser.urlencoded({ extended: false }));
app.use(cookieParser());
app.use(express.static(path.join(__dirname, 'public')));

app.use('/', routes);

// catch 404 and forward to error handler
```




```
function d() { ... };
function c() { ...; setTimeout(d, 200); };
function b() { c(); };
function a() { b(); };
a();
```

Non-blocking

- An algorithm is called non-blocking if failure/ suspension of any thread cannot cause failure/suspension of another thread

Simplest Case

```
$.getJSON('http://example.com/posts',
          (result) => console.log(result));
```

http://wudi.link/568EEC

Categories of Async Fns

- setTimeout/setInterval
- I/O
- …

Callback Hell

```
$.getJSON('http://lzx.com/posts', (result) => {
  $.getJSON('http://lzx.com/users', (result) => {
    $.getJSON('http://lzx.com/features', (result) => {
    });
  });
});
```

Simple Solution

```
$.getJSON('http://lzx.com/pics', (result) => {
  // do something
  anotherCall();
  // do something else
});

function anotherCall() {
  $.getJSON('http://lzx.com/cars', (result) => {
    // ...
  });
}
```

http://wudi.link/E69412

Error Handling Caveat

- When error is thrown asynchronously, it cannot retain the original call stack

- http://wudi.link/3C3F0b

Asynchronous JavaScript Patterns

Lecture 2

Async.js

- Build async workflow
- Error handling
- http://wudi.link/F0C0B1

PubSub

- Publisher
- Subscriber

```
el.addEventListener('click', function(e) {
  // handle click event
});
```

jQuery Custom Events

- Works on any object
- Events can be customized (http://wudi.link/5d0439)
- Implement your own Custom Event System (http://wudi.link/b5e40B)

```
// subscribe
$("#foo").on("click", function() {
  alert($(this).text());
});

// publish
$("#foo").trigger("click");
```

Caveat

```
$(dom).on('click', function() {
  $(this).trigger('click');
  // Call stack overflow!
});
```

Error Handling

- Pub/Sub to `error` event

Promise

```
doSomething(function(value) {
  console.log('Got a value:' + value);
});
```

```
doSomething().then(function(value) {
  console.log('Got a value:' + value);
});
```

```
function doSomething() {
  return {
    then: function(callback) {
      var value = 42;
      callback(value);
    }
  };
}
```

Promise

- Intuition (http://wudi.link/d70aa0)
- Promise Chaining (http://wudi.link/420b42)
- catch

Promise

- No more callback hell (yay!)
- We can now use **throw/catch** and **return** in async code!

Promise

- *We have a problem with promise* **(***http://wudi.link/883605***)**

ES7 async/wait

```
async function loadStory() {
  try {
    let story = await getJSON('story.json');
    addHtmlToPage(story.heading);
    for (let chapter of story.chapterURLs.map(getJSON)) {
      addHtmlToPage((await chapter).html);
    }
    addTextToPage("All done");
  } catch (err) {
    addTextToPage("Argh, broken: " + err.message);
  }
}
```

Reactive Programming

Lecture 3

JS - Functional Part

- forEach
- map
- reduce
- filter
- concatAll (not native JS)
- Functional Programming in JS (http://wudi.link/b60ceA)

Think of A Mouse


…(time)…

…(time)…


Mouse Click Event Stream

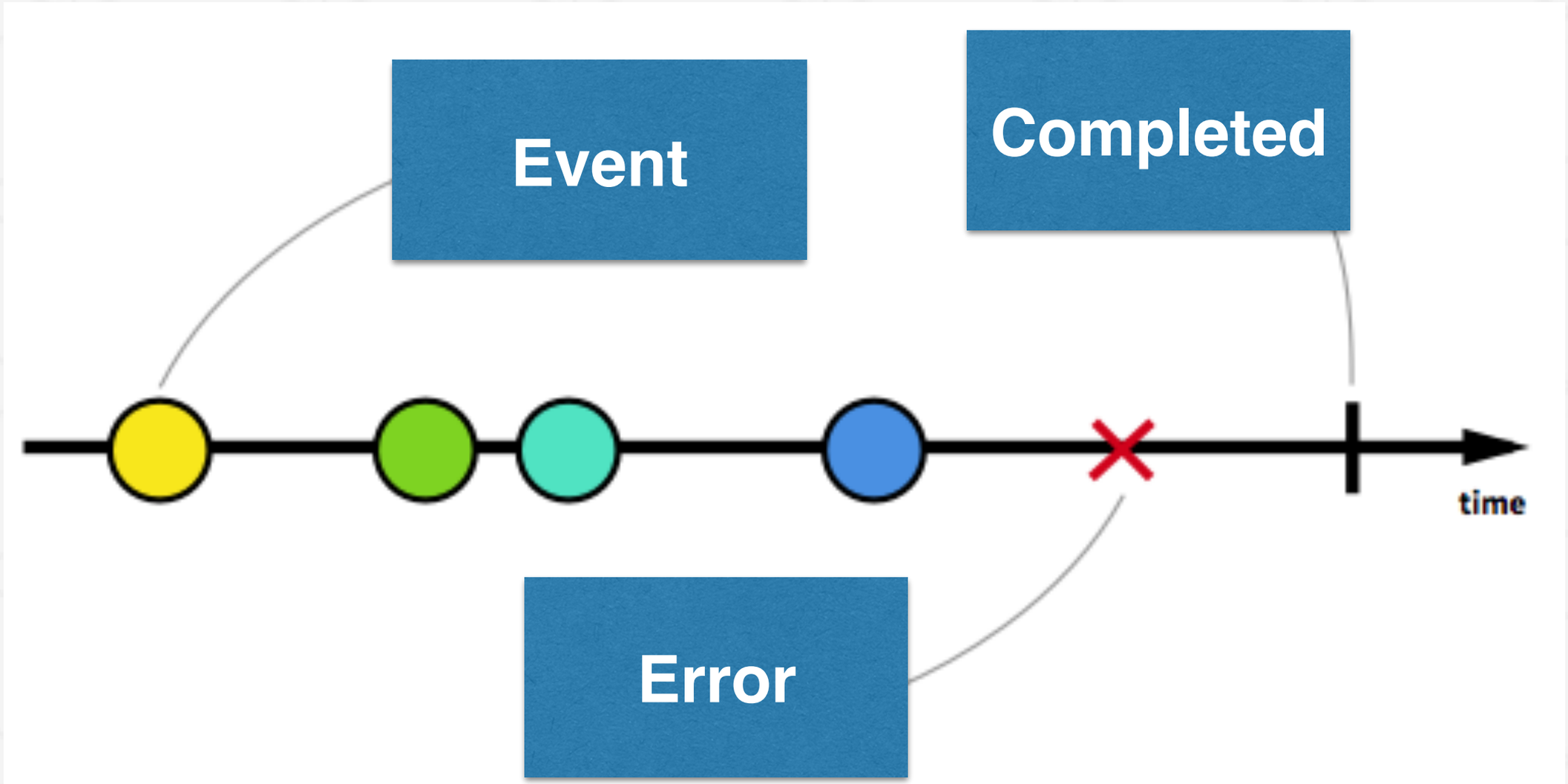


Image Credit: http://wudi.link/79242B

Event Stream

vs.

Array

Observer Pattern

vs.

Iterator Pattern

```
mouseClicks.forEach(function(e) {
  // process click event
});
```

Iterator Pattern on Streams!

```
var getElementDrags = elmt =>
    elmt.mouseDowns.
        map(mouseDown =>
            document.mouseMoves.
                takeUntil(document.mouseUps)).
        concatAll();

getElementDrags(image).
    forEach(pos => image.position = pos);
```

Drag & Drop!

RxJS

- Observable
- Interactive diagram of Observables (http://wudi.link/037693)

Demo

- Drag & Drop (http://wudi.link/FD8571)

Wrap Up

A Final Touch

Keywords

- Internal
- Patterns & Tools
- Scenario

Going Further

- Jafar Husain: Async Programming in ES7 (https://www.youtube.com/watch?v=lil4YCCXRYc)
- The introduction to Reactive Programming you've been missing (https://gist.github.com/staltz/868e7e9bc2a7b8c1f754)
- JavaScript Promises - There and back again (http://www.html5rocks.com/zh/tutorials/es6/promises/)