



Web Development

COMP 431 / COMP 531

Lecture 8: Promises and Fetch

Instructor: Mack Joyner
Department of Computer Science, Rice University

mjoyner@rice.edu

<http://www.clear.rice.edu/comp431>

Recap

- HTML and HTML5, Storage, Canvas
- JavaScript and Scope
- Forms
- CSS
https://www.clear.rice.edu/comp431/pdfs/lec_css.pdf
- Events

Quiz 1: JavaScript

Due Sunday 9/24

*Homework Assignment 3
(JS Game)*

Due Thursday 9/28

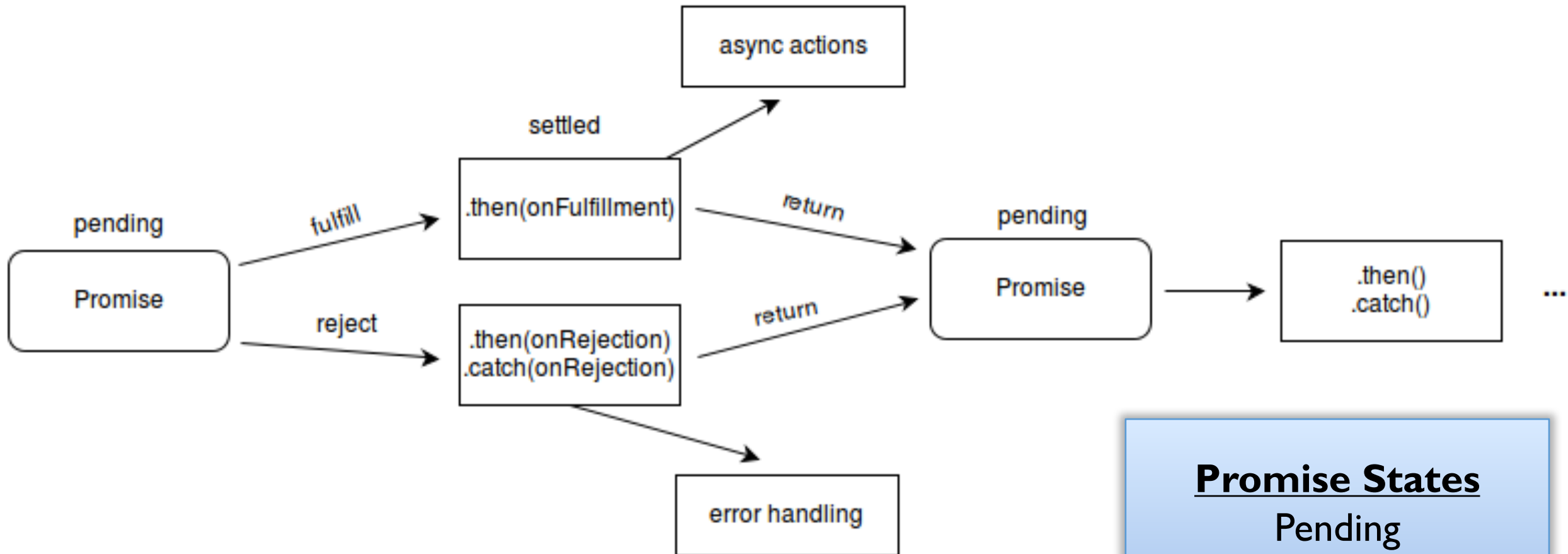
COMP 531 Final Presentations

Thursday 11/30, 12/7

jQuery Callback vs Chaining

```
$("#div3").click(function() {  
    $(this)  
        .animate( {opacity: 0}, 2500)  
        .animate( {opacity: 1, fontSize: '1em' }, 500 )  
        .hide(1000, function() {  
            $(this).css({ backgroundColor: "blue" })  
        })  
        .show(1000)  
        .animate( { fontSize: '2em' }, function() {  
            $(this).css({ backgroundColor: "green" })  
        } )  
    })
```

Better than Callbacks => Promises



Promise States

Pending
Fulfilled / Rejected
Settled

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Promise

See also <http://www.html5rocks.com/en/tutorials/es6/promises/>

Communicating with the Server

- GET
 - POST
 - ...
-
- Always instantiated by the browser (i.e., user) perhaps using a link or button (e.g., to submit a form)
 - We'd like to have JavaScript control to ask the Server for data

JavaScript Requests



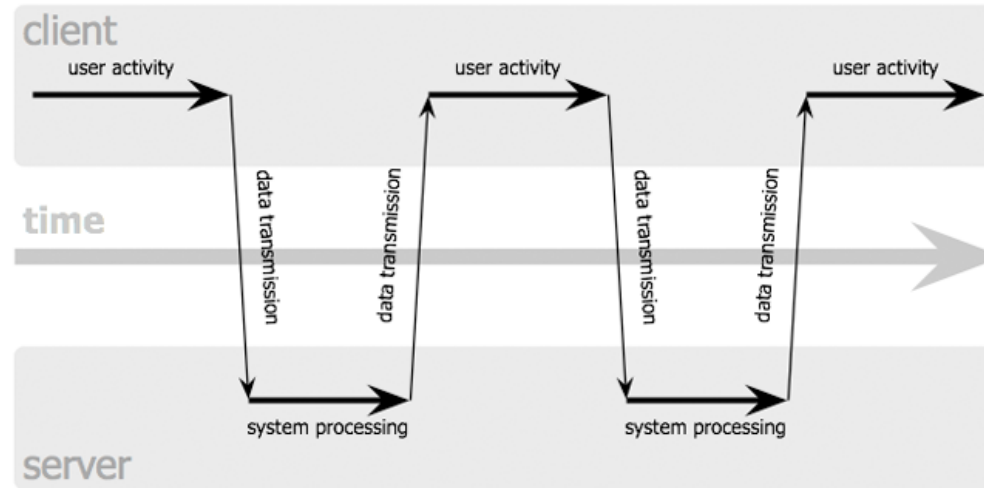
```
var url = 'http://webdev-dummy.herokuapp.com/sample';
console.log('Make request to ', url);
var req = new XMLHttpRequest();
req.open('GET', url);
req.onload = function() {
    console.log('Request status', req.status);
    console.log('Response size', req.response.toString().length);
}
req.send();
```

Request status 200

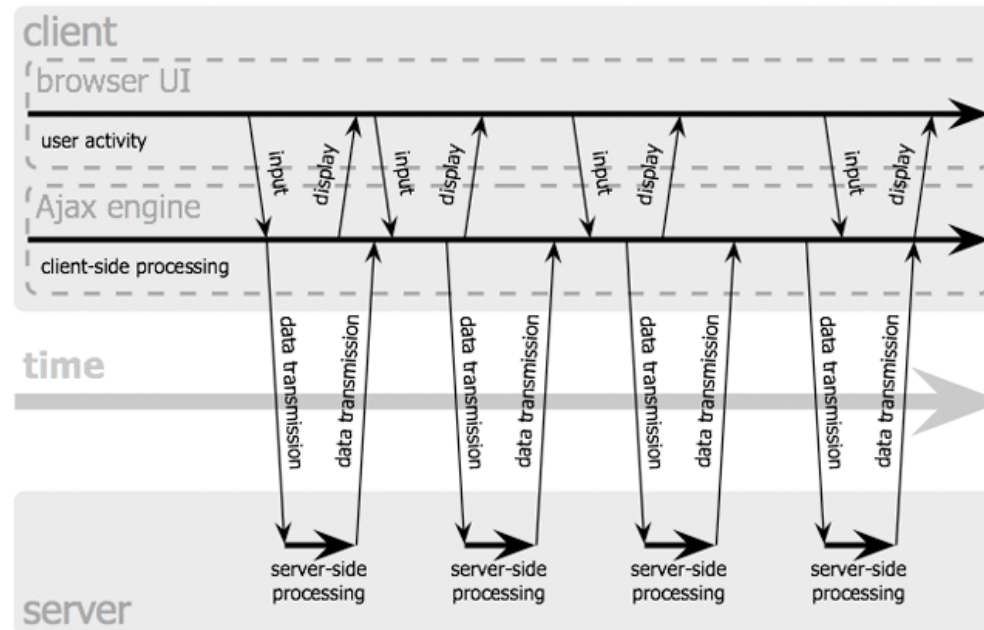
Response size 6960

The “a” in AJAX





classic web application model (synchronous)



Ajax web application model (asynchronous)



Event Timeline

Name	×	Headers	Preview	Response	Cookies	Timing
 hello-jquery.html	Connection Setup					TIME
 hello-jquery.js	Queueing					0.759 ms
 jquery.min.js	Stalled		█			6.746 ms
 api/	DNS Lookup					1.046 ms
	Initial connection		<div></div>			200.667 ms
	Request/Response					TIME
	Request sent					0.522 ms
	Waiting (TTFB)		<div></div>			509.809 ms
	Content Download				<div></div>	17.496 ms
4 requests 31.9 KB t...						Explanation 737.372 ms

The Fetch API

Fetch is NATIVE !!

```
> fetch
< function fetch() { [native code] }
> fetch('https://webdev-dummy.herokuapp.com/sample')
  .then(r => r.json() )
  .then(r => console.log(r) )
< ▶ Promise {[[PromiseStatus]]: "pending", [[PromiseValue]]: undefined}
```

**Inline function callbacks
with arrow notation**

VM126:3

```
▼ Object {articles: Array[15]} ⓘ
  ▼ articles: Array[15]
    ▼ 0: Object
      _id: 4532479
      author: "ral8"
      ▶ comments: Array[0]
      date: "2015-08-24T22:39:18.729Z"
      img: null
      text: "Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aenean co
```

Same as:

```
.then(function(r) {
  return r.json()
})
r.json() returns a Promise, the next
then() is called when json() resolves.
```

The Fetch API

set options, such as request headers,
request method, payload

Check response headers

text() and json() are promises

Create an exception with
the text message

catch the exception

```
const url = 'https://webdev-dummy.herokuapp.com'
fetch(`${url}/foobar`, {
  headers: {
    'Content-Type': 'application/json'
  }
})
  .then(r => {
    const contentType = r.headers.get('Content-Type')
    if (contentType.indexOf('application/json') >= 0) {
      console.log('We received JSON!')
      return r.json()
    } else {
      console.log("We did NOT receive JSON!")
      return r.text().then(msg => {
        throw new Error(msg)
      })
    }
  })
  .then(r => console.log(Object.keys(r)))
  .catch(err => console.error('There was a problem', err))
```

JavaScript Object Notation (JSON)

Be careful when building string to convert to JSON



Client and Server use JSON to communicate data to/from persistent storage (e.g. database)

```
> total = JSON.parse('{age:30}')
```

```
✖ ▶ Uncaught SyntaxError: Unexpected token a in      VM412:1  
  JSON at position 1  
    at JSON.parse (<anonymous>)  
    at <anonymous>:1:14
```

```
> total = JSON.parse('{"age":30}')
```

```
◀ ▶ {age: 30}
```

JavaScript Object Notation (JSON)

Object.keys(), Object.values()
work for JSON

Error occurs when passing Object.keys()
a non-JSON argument



```
function parseJSON(url) {  
  
  //print out the JSON key, value pairs (assuming input is json here!)  
  return fetch(url)  
    .then(res => res.json())  
    .then(res => console('keys: ' + Object.keys(res) + ', values: ' + Object.values(res)));  
}
```

JSON (in-class exercise) Error

Might be a JSON parse issue,
check the JavaScript console

passes: 0 failures: 1 duration: 2.02s 0%

Mocha+Chai Inclass Fetch Exercise for "Mack Joyner"

✖ "before each" hook for "author should be defined"

Error: timeout of 2000ms exceeded. Ensure the
at <https://cdnjs.cloudflare.com/ajax/libs>.


Mocha and Chai

- Mocha is framework to create tests
 - Uses Chai for test assertions
- Chai is a behavior and test driven development assertion library

```
<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/mocha/3.0.0-2/mocha.min.css">
<script src="https://cdnjs.cloudflare.com/ajax/libs/mocha/3.0.0-2/mocha.min.js"></script>
<script src="https://cdnjs.cloudflare.com/ajax/libs/chai/3.5.0/chai.min.js"></script>
```


Mocha and Chai

Mocha test description



```
describe(`Mocha+Chai Inclass Fetch Exercise for "${inclass.author}"`, () => {  
  const baseURL = 'https://webdev-dummy.herokuapp.com'  
  const sample = `${baseURL}/sample`
```

Mocha test to run



```
it('author should be defined', () => {  
  expect(inclass.author).to.be.ok  
})
```

Chai test assertion

