“Housekeeping”

- Welcome to today’s ACM Webinar. The presentation starts at the top of the hour.

- If you are experiencing any problems/issues, refresh your console by pressing the F5 key on your keyboard in Windows, Command + R if on a Mac, or refresh your browser if you’re on a mobile device; or close and re-launch the presentation. You can also view the Webcast Help Guide, by clicking on the “Help” widget in the bottom dock.

- To control volume, adjust the master volume on your computer.

- If you think of a question during the presentation, please type it into the Q&A box and click on the submit button. You do not need to wait until the end of the presentation to begin submitting questions.

- At the end of the presentation, you’ll see a survey open in your browser. Please take a minute to fill it out to help us improve your next webinar experience.

- You can download a copy of these slides by clicking on the Resources widget in the bottom dock.

- This presentation is being recorded and will be available for on-demand viewing in the next 1-2 days. You will receive an automatic e-mail notification when the recording is ready.
Async JavaScript at Netflix

Jafar Husain
@jhusain
ACM Learning Center
http://learning.acm.org

- 1,400+ trusted technical books and videos by leading publishers including O’ Reilly, Morgan Kaufmann, others

- Online courses with assessments and certification-track mentoring, member discounts on tuition at partner institutions

- Learning Webinars on big topics (Cloud/Mobile Development, Cybersecurity, Big Data, Recommender Systems, SaaS, Agile, Machine Learning, NLP, Hadoop Parallel Programming, etc.)

- ACM Tech Packs on top current computing topics: Annotated Bibliographies compiled by subject experts

- Popular video tutorials/keynotes from ACM Digital Library, A.M. Turing Centenary talks/panels

- Podcasts with industry leaders/award winners
“Housekeeping”

- Welcome to today’s ACM Webinar. The presentation starts at the top of the hour.

- If you are experiencing any problems/issues, refresh your console by pressing the **F5** key on your keyboard in **Windows**, **Command + R** if on a **Mac**, or refresh your browser if you’re on a mobile device; or close and re-launch the presentation. You can also view the Webcast Help Guide, by clicking on the “Help” widget in the bottom dock.

- To control volume, adjust the master volume on your computer.

- If you think of a question during the presentation, please type it into the **Q&A** box and click on the submit button. You do not need to wait until the end of the presentation to begin submitting questions.

- At the end of the presentation, you’ll see a **survey** open in your browser. Please take a minute to fill it out to help us improve your next webinar experience.

- You can download a copy of these slides by clicking on the **Resources** widget in the bottom dock.

- This presentation is being recorded and will be available for on-demand viewing in the next 1-2 days. You will receive an **automatic e-mail notification** when the recording is ready.
Talk Back

• Use the Facebook widget in the bottom panel to share this presentation with friends and colleagues

• Use Twitter widget to Tweet your favorite quotes from today’s presentation with hashtag #ACMWebinarAsyncJS

• Submit questions and comments via Twitter to @acmeducation – we’re reading them!
Who is Jafar?

- Cross-Team Technical Lead for the Netflix UIs
- Created the async data platform for Netflix UI’s
- Member of TC39
- 13 years in the industry, formerly worked at Microsoft and GE
This is the story of how Netflix solved BIG async problems by thinking differently about Events.
The Netflix App is Asynchronous

- App Startup
- Player
- Data Access
- Animations
- View/Model binding
Async Problems

- Memory Leaks
- Race Conditions
- Callback Hell
- Complex state machines
- Error Handling
Async is Hard

```javascript
function play(movieId, cancelButton, callback) {
  var movieTicket,
      playError,
      tryFinish = function() {
        if (playError) {
          callback(null, playError);
        } else if (movieTicket && player.initialized) {
          callback(null, ticket);
        }
      };
  cancelButton.addEventListener("click", function() { playError = "cancelled"; }
  if (!player.initialized) {
    player.init(function(error) {
      playError = error;
      tryFinish();
    });
  }
  authorizeMovie(function(error, ticket) {
    playError = error;
    movieTicket = ticket;
    tryFinish();
  });
}
```
Design Pattern Relationships
var iterator = getNumbers();
console.log(iterator.next());
{ value: 1, done: false }
console.log(iterator.next());
{ value: 2, done: false }
console.log(iterator.next());
{ value: 3, done: false }
console.log(iterator.next());
{ done: true }
Observer Pattern

```javascript
> document.addEventListener(
  "mousemove",
  function next(e) {
    console.log(e);
  });
> { clientX: 425, clientY: 543 }
> { clientX: 450, clientY: 558 }
> { clientX: 455, clientY: 562 }
> { clientX: 460, clientY: 743 }
> { clientX: 476, clientY: 760 }
> { clientX: 476, clientY: 760 }
> { clientX: 476, clientY: 760 }
> { clientX: 476, clientY: 760 }
```
Iterator progressively send information to consumer Observer
IN OBSERVER PATTERN

PRODUCER ITERATE YOU
“What’s the difference between an Array…

[\{x: 23, y: 44\}, \{x:27, y:55\}, \{x:27, y:55\}]
... and an Event?
Events and Arrays are *both* collections.
Now for a brief JavaScript 6 tutorial...
Functions

function(x) { return x + 1; }
function(x, y) { return x + y; }
Fin.
The majority of Netflix’s async code is written with just a few flexible functions.
ForEach

> [1, 2, 3].forEach(x => console.log(x))
> 1
> 2
> 3
Map
Map

> [1, 2, 3].map(x => x + 1)  
> [2, 3, 4]

>
Filter
Filter

> \[1, 2, 3\].filter(x => x > 1)
> \[2, 3\]
> \[\]
concatAll
concatAll

> [ [1], [2, 3], [], [4] ].concatAll()
> [1, 2, 3, 4]
Map/Filter/ConcatAll

> `[1, 2, 3].map(x => x + 1)`
> `[2, 3, 4]`

> `[1, 2, 3].filter(x => x > 1)`
> `[2, 3]`

> `[ [1], [2, 3], [], [4] ].concatAll()`
> `[1, 2, 3, 4]`
Orange is the New Black

2013  TV-MA  13 episodes

From the creator of “Weeds” comes this series about a privileged New Yorker who ends up in a women’s prison when a past crime catches up with her.

Based on your interest in: Breaking Bad
Let’s use `map`, `filter`, and `concatAll` to get a list of your favorite Netflix titles.
Top-rated Movies Collection

```javascript
var getTopRatedFilms = user =>
  user.videoLists.
    map(videoList =>
      videoList.videos.
        filter(video => video.rating === 5.0)).
    concatAll();

getTopRatedFilms(user).
  forEach(film => console.log(film));
```
What if I told you...

...that you could create a drag event...

...with nearly the same code?
var getTopRatedFilms = user =>
    user.videoLists.
        map(videoList =>
            videoList.videos.
                filter(video => video.rating === 5.0)).
        concatAll();

getTopRatedFilms(user).
    forEach(film => console.log(film));
Mouse Drags Collection

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

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

Observable === Collection + Time
Reactive Extensions

- Observable Type + Array Functions (and more)
- Open Source
- Ported to…
  - C
  - C#/VB.Net
  - Javascript
  - Java (Netflix)
Observables can model…

- Events
- Animations
- Async IO
Events to Observables

```javascript
var mouseMoves = Observable.fromEvent(element, "mousemove");
```
Event Subscription

// “subscribe”
var handler = (e) => console.log(e);
document.addEventListener("mousemove", handler);

// “unsubscribe”
document.removeEventListener("mousemove", handler);
Observable.forEach

// “subscribe”
var subscription = mouseMoves.forEach(console.log);

// “unsubscribe”
subscription.dispose();
Expanded Observable.forEach

// “subscribe”
var subscription =
  mouseMoves.forEach(
    // next data
    event => console.log(event),
    // error
    error => console.error(error),
    // completed
    () => console.log("done");

// “unsubscribe”
subscription.dispose();
Observable Literal

$\{1 \ldots 2 \ldots \ldots \ldots \ldots \ldots \ldots \ldots 3\}$
ForEach

```javascript
> {1......2............3}.forEach(console.log)
> 1
> 2
> 3
```
Map

> {1....2..........3}.map(x => x + 1)
> 2
> 3
> 4
Filter

```javascript
> {1...2...3}.filter(x => x + 1)
```

```
> [2, 3]
```

**time**
concatAll

[1]
[2, 3],
[],
[4]
].concatAll()

[1, 2, 3, 4]
concatAll

{...
...{1}
......{2-----------3},
.........{} 
.........{4} 
}.concatAll()

{...1...2-----------3...4}
mergeAll

time

{...
  {1}
  {2........3},
  {}
  {4}
}.mergeAll()

{...1...2...4......3}
switchLatest

```javascript
{ ...{1}
    ....{2...............3},
    ..........{ }
    ..........{4}
}.switchLatest()

subscription.dispose()
```
TakeUntil

\{\ldots1\ldots2\ldots\ldots3\}\).takeUntil(\{\ldots\ldots4\}\)
Don’t unsubscribe from Events. Complete them when another event fires.
Mouse Drags Collection

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

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

- Prison Break
- Peep Show
- Pirates of the Caribbean: The Curse of the Black Pearl
Netflix Search

```javascript
var searchResultSets = keyPresses.
  .throttle(250).
  .map(key =>
    getJSON(`/searchResults?q=" + input.value).
    .retry(3).
    .takeUntil(keyPresses)).
  .concatAll();

searchResultSets.forEach(
  resultSet => updateSearchResults(resultSet),
  error => showMessage("the server appears to be down."));
```
var searchResultSets = keyPresses.
  .throttle(250).
  .map(key =>
    getJSON(“/searchResults?q=” + input.value).
    .retry(3).
    .takeUntil(keyPresses)).
  .concatAll switchLatest();

searchResultSets.forEach(
  resultSet => updateSearchResults(resultSet),
  error => showMessage(“the server appears to be down.”));
var searchResultSets = keyPresses.
    throttle(250).
    map(key =>
        getJSON("/searchResults?q=" + input.value).
            retry(3)).
    switchLatest();

searchResultSets.forEach(
    resultSet => updateSearchResults(resultSet),
    error => showMessage("the server appears to be down.");
Netflix Player
function play(movieId, cancelButton, callback) {
    var movieTicket,
        playError,
        tryFinish = function() {
            if (playError) {
                callback(null, playError);
            } else if (movieTicket && player.initialized) {
                callback(null, ticket);
            }
        }
    cancelButton.addEventListener("click", function() { playError = "cancel"; });
    if (!player.initialized) {
        player.init(function(error) {
            playError = error;
            tryFinish();
        });
    }
    authorizeMovie(movieId, function(error, ticket) {
        playError = error;
        movieTicket = ticket;
        tryFinish();
    });
}
Player with Observable

```javascript
var authorizations =
    player.
    init().
    map(() =>
        playAttempts.
            map(movieId =>
                player.authorize(movieId).
                    catch(e => Observable.empty).
                        takeUntil(cancels)).
                concatAll()).
        concatAll());

authorizations.forEach(
    license => player.play(license),
    error => showDialog("Sorry, can’t play right now.");
```
Netflix: Observable Everywhere

- App Startup ✔
- Player ✔
- Data Access ✔
- Animations ✔
- View/Model binding ✔
Interactive Learning Exercises

http://jhusain.github.io/learnrx/
Observable in JavaScript 7?

```javascript
async function* getStocks() {
    let reader = new AsyncFileReader("stocks.txt");
    try {
        while(!reader.eof) {
            let line = await reader.readLine();
            await yield JSON.parse(line);
        }
    }
    finally {
        reader.close();
    }
}

async function writeStockInfos() {
    let writer = new AsyncFileWriter("stocksAndPrices.txt");
    try {
        for(let name on getStocks()) {
            let price = await getStockPrice(name);
            await writer.writeLine(JSON.stringify({name, price}));
        }
    }
    finally {
        writer.close();
    }
}
```
Resources

- reactivetrader.azurewebsites.net
- https://github.com/Reactive-Extensions/RxJS
- RxJava
- http://jhusain.github.io/learnrx/
- @jhusain
Questions
ACM: The Learning Continues…

- Questions about this webcast? learning@acm.org

- ACM Learning Webinars (on-demand archive): http://learning.acm.org/webinar

- ACM Learning Center: http://learning.acm.org

- ACM Queue: http://queue.acm.org/