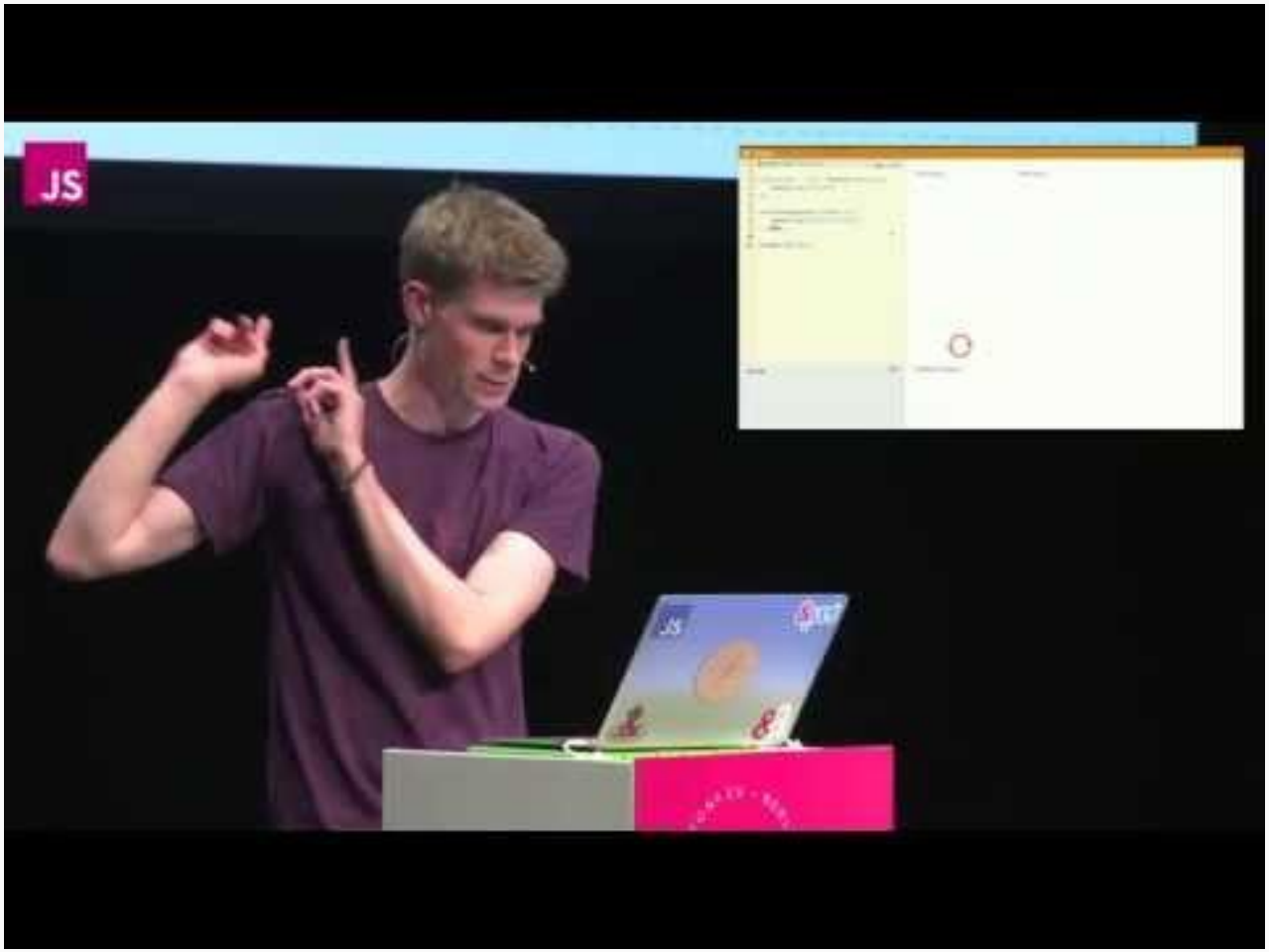
CSC309 - Winter 2017

Lab 9 - Understanding JS Event Loops, Scopes and JSONP

JS Event Loops



- TAs please skim through the important parts of this video, answer any questions students might have.
- Demo Tool: http://latentflip.com/loupe/

Scope

```
var school = ‘Uoft’

function getSchoolName() {
    var campus = ‘SG’;
    return school + ‘-’ + campus;

    // code here can use campus and school vars;
}

// code here can not use campus
console.log(getSchoolName());
```

- School variable is in the global scope.
- Campus variable is limited to the getSchoolName function scope.

Scopes and Namespaces

```
// namespace
var SchoolApp = {};

// property
SchoolApp.name = 'UofT';

/**
 * Method to get school name;
 *
 * @return schoolname.
 */
SchoolApp.getSchoolName = function() {
    var campus = 'SG';
    return this.name + '-' + campus;
}
```

- The aim of using namespace is to not litter the global namespace with our variables. This can be problematic if you're loading multiple JS files which also do not have namespacing. You will end up overriding properties of each other
- Name property is in global scope within the SchoolApp namespace.
- Campus variable is limited to the getSchoolName method.
- 'this' always represents the current scope and whatever belongs to it.

Scopes and Namespaces

```
// namespace
var SchoolApp = {};

// property
SchoolApp.name = 'UofT';

/**
 * Method to get school name;
 *
 * @return schoolname.
 */
SchoolApp.getSchoolName = function() {
    var campus = 'SG';
    return this.name + '-' + campus;
}

SchoolApp.init = function() {
    console.log(this.getSchoolName());
}

SchoolApp.init();
```

```
// namespace
var SchoolApp = {};

// property
SchoolApp.name = 'UofT';

/**
 * Method to get school name;
 *
 * @return schoolname.
 */
SchoolApp.getSchoolName = function() {
    var campus = 'SG';
    return this.name + '-' + campus;
}

SchoolApp.init = function() {
    setTimeOut(this.getSchoolName, 1);
}

SchoolApp.init();
```

- The code on the left works but the one on the right fails.
- Spot the difference.
- getSchoolName method is invoked by setTimeOut Function, what does this mean?

Scopes and Namespaces

```
// namespace
var SchoolApp = {};

// property
SchoolApp.name = 'UofT';

/**
 * Method to get school name;
 *
 * @return schoolname.
 */
SchoolApp.getSchoolName = function() {
    var campus = 'SG';
    return this.name + '-' + campus;
}

SchoolApp.init = function() {
    console.log(this.getSchoolName());
}

SchoolApp.init();
```

```
// namespace
var SchoolApp = {};

// property
SchoolApp.name = 'UofT';

/**
 * Method to get school name;
 *
 * @return schoolname.
 */
SchoolApp.getSchoolName = function() {
    var campus = 'SG';
    return this.name + '-' + campus;
}

SchoolApp.init = function() {
    setTimeOut(this.getSchoolName, 1);
}

SchoolApp.init();
```

- Scopes change depending on how a method is invoked.
- In the left the scope was bound to the SchoolApp namespace.
- On the right the scope is bound to the setTimeOut event, you can change this, but you need to figure out how. (this is a code lab task).

JsonP - JSON with Padding

- Consider a scenario where you need to pull in information from a JSON api.
- It's common to fetch information from the api hosted on the same domain.
- What happens when you try to fetch information from other domains.
- This is a security flaw, this violates the 'SAME-ORIGIN' policy which is implemented by browsers to prevent XSS (Cross Site Scripting) attacks.
- JSONP helps implement CORS (Cross Origin Resource Sharing)
- As the name suggest, we can host the same API on one domain and access it from any website.
- e.g. if we want to access instagram's images, you cannot simply pull in the posts as JSON, for this purpose they provide an sdk, which implements jsonp at some level.
- The aim is to provide data padded as a function which can be executed in the code.

```
function jsonp(url, callback) {
 var callbackName = 'foo';
 var script =
document.createElement('script');
 script.src = url +
(url.indexOf('?') >= 0 ? '&' : '?')
+ 'callback=' + callbackName;
 document.body.appendChild(script);
};

// This code basically loads a jsonp
resource as a script and executes
it. Everything in the script becomes
available in the globalScope.

Demo:
https://instareproxy.herokuapp.com/u
oft/media/?count=5&callback=foo

ASK TAs if SOMETHING IS NOT CLEAR!
```

Quiz Time