



Web Application Development with Closure Compiler

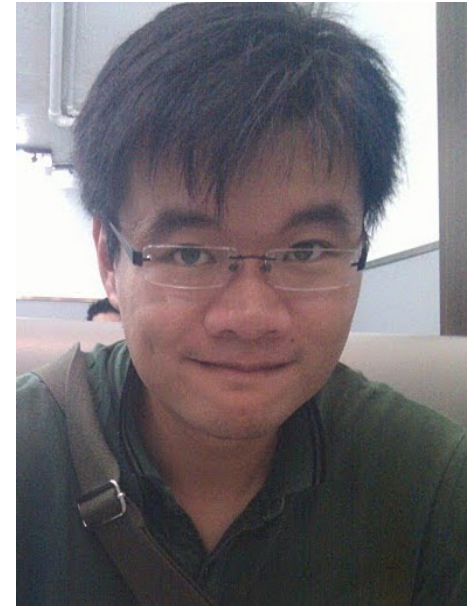
Alan Leung



About me:

Alan Leung

- Closure Compiler Tech Lead at Google
- University of Waterloo:
JavaVM, Graphics Card Shader Compilers
- `acleung [at] gmail.com / google.com`
- Twitter `@acleung`
- These slides can be found at http://acleung.com/twitter_talk.pdf

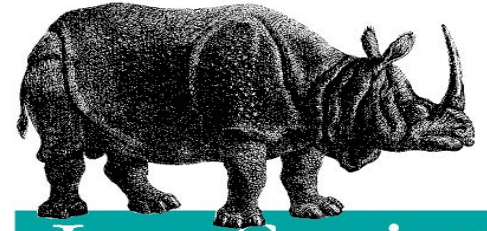


Outline

Closure Compiler:

- What is Closure Compiler?
- Using Closure Compiler to speed up development
- Using Closure Compiler to optimize code
- Other Tips and Tricks
- Conclusion / Questions

Background



JavaScript

- Source-to-source Javascript compiler
- Started in 2004, as a weekend experiments processor for GMail javascript using Rhino's parser.
- Mainly 20% contribution until 2008
- Industry strength production compiler with over 30+ optimization pass



Who uses it?



- Almost every Google web frontend uses it: GMail, Calendar, Maps, Search, Docs, Spreadsheets, Blogger, Reader, Books...
- Opensourced 2009: Maven, Ant, Visual Studio.. integration, Coffeescript, Emscripten
- Around 600 downloads last month's release
- <http://closure-compiler.appspot.com/>
~0.5 compiles per second

Goals

- 1) Speed up development process
- 2) Speed up web applications

Speed up development

- Javascript was initially used for making small modification to the HTML DOM.

Speed up development

- Javascript was initially used for making small modification to the HTML DOM.
- Javascript is an **AWESOMELY** dynamic which lets you modify anything: monkey patching class, dynamically convert types, variable are allocate on the fly

Speed up development

- Javascript was initially used for making small modification to the HTML DOM.
- Javascript is an **AWESOMELY** dynamic which lets you modify anything: monkey patching class, dynamically convert types, variable are allocate on the fly
- Javascript is an **AWFULLY** dynamic which lets you modify anything

Speed up development

- GMail at some point has over 200 engineers...
- Web applications are doing a lot of advanced features: dynamic module loading, Lab features, i18n
- Use Closure Compiler to add and refine certain things in the language

Demonstration



Web App

The screenshot shows the Closure Compiler web application interface in a Chrome browser. The browser's address bar displays the URL `http://closure-compiler.appspot.com/home`. The page title is "Closure Compiler".

On the left side, there is a form for adding a URL. The "Add a URL:" field contains `http://`. Below it is an "Add" button and an example URL: `http://www.example.com/bigfile.js`. The "Optimization:" section has three radio buttons: "Whitespace only", "Simple" (which is selected), and "Advanced". A link [Which optimization is right for my code?](#) is provided. The "Formatting:" section has two checkboxes: "Pretty print" and "Print input delimiter". There are "Compile" and "Reset" buttons.

The main area on the right is for the compilation results. It shows "Original Size:" and "Compiled Size:" fields. Below these are four tabs: "Compiled Code", "Warnings", "Errors", and "POST data". The "Compiled Code" tab is currently selected and is empty.

At the bottom of the page, there is a copyright notice: ©2009 Google - [Terms of Service](#) - [Privacy Policy](#) - [Google Home](#).

```
// ==ClosureCompiler==
// @compilation_level SIMPLE_OPTIMIZATIONS
// @output_file_name default.js
// ==/ClosureCompiler==

// ADD YOUR CODE HERE
function hello(name) {
  alert('Hello, ' + name);
}
hello('New user');
```



Web App



- Modify the Javascript to display different information in some internal version

Web App



- Modify the Javascript to display different information in some internal version
- In particular, we want to display the GET URL of the webservice request when user select the POST data tab

Web App



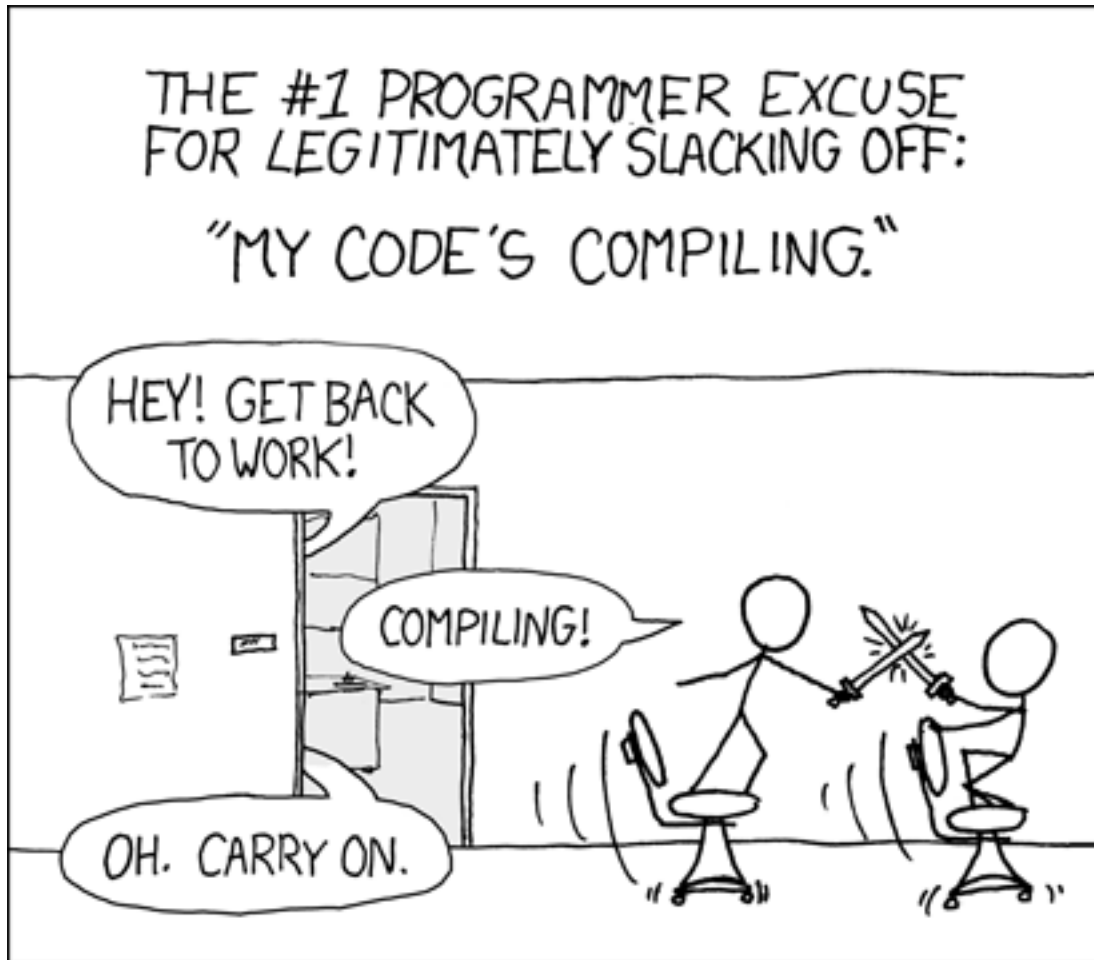
- Modify the Javascript to display different information in the dogfood version
- In particular, we want to display the GET URL of the webservice request when user select the POST data tab
- That's like 5 minutes worth of Javascript code!

Javascript Hacking

```
1 /**
2  * @param {string} x
3  */
4 Request.prototype.getData = function (x) {
5   return
6   [ this.getUrl(), this.getParam(), this.getDate(), ].join(x);
7 }
8
9 /**
10 * @return {string}
11 */
12 giffy.render = function () {
13   var x = Respond();
14   var separator;
15   if (!INTERNAL) {
16     separator = "&";
17     return "";
18   } else {
19     return x.getData(separator);
20   }
21 }
```



Compiling with Closure Compiler



<http://xkcd.com/303/>

Compiling with Closure Compiler

ERROR - Parse error. Internet Explorer has a non-standard interpretation of trailing commas. Arrays will have the wrong length and objects will not parse at all.

```
[this.getUrl(), this.getParam(), this.getDate(),].join(x);
```



Javascript Hacking

```
1 /**
2  * @param {string} x
3  */
4 Request.prototype.getData = function (x) {
5   return
6   [ this.getUrl(), this.getParam(), this.getDate() ].join(x);
7 }
8
9 /**
10 * @return {string}
11 */
12 giffy.render = function () {
13   var x = Respond();
14   var separator;
15   if (!INTERNAL) {
16     separator = "&";
17     return "";
18   } else {
19     return x.getData(separator);
20   }
21 }
```



Compiling with Closure Compiler

ERROR - Constructor function (this:Respond): ? should be called with the "new" keyword
var x = Respond();



Javascript Hacking

```
1 /**
2  * @param {string} x
3  */
4 Request.prototype.getData = function (x) {
5   return
6   [ this.getUrl(), this.getParam(), this.getDate() ].join(x);
7 }
8
9 /**
10 * @return {string}
11 */
12 giffy.render = function () {
13   var x = new Respond();
14   var separator;
15   if (!INTERNAL) {
16     separator = "&";
17     return "";
18   } else {
19     return x.getData(separator);
20   }
21 }
```



Compiling with Closure Compiler

a.js:19: ERROR - Property getData never defined on Respond
return x.getData(separator);



Javascript Hacking

```
1 /**
2  * @param {string} x
3  */
4 Request.prototype.getData = function (x) {
5   return
6   [ this.getUrl(), this.getParam(), this.getDate() ].join(x);
7 }
8
9 /**
10 * @return {string}
11 */
12 giffy.render = function () {
13   var x = new Respond();
14   var seperator;
15   if (!INTERNAL) {
16     seperator = "&";
17     return "";
18   } else {
19     return x.getData(seperator);
20   }
21 }
```



Javascript Hacking

```
1 /**
2  * @param {string} x
3  */
4 Respond.prototype.getData = function (x) {
5   return
6   [ this.getUrl(), this.getParam(), this.getDate() ].join(x);
7 }
8
9 /**
10 * @return {string}
11 */
12 giffy.render = function () {
13   var x = new Respond();
14   var separator;
15   if (!INTERNAL) {
16     separator = "&";
17     return "";
18   } else {
19     return x.getData(separator);
20   }
21 }
```



Compiling with Closure Compiler

a.js:19: ERROR - actual parameter 1 of Respond.prototype.getData does not match formal parameter

found : undefined

required: string

```
return x.getData(separator);
```



Javascript Hacking

```
1 /**
2  * @param {string} x
3  */
4 Request.prototype.getData = function (x) {
5   return
6   [ this.getUrl(), this.getParam(), this.getDate() ].join(x);
7 }
8
9 /**
10 * @return {string}
11 */
12 giffy.render = function () {
13   var x = new Respond();
14   var separator;
15   if (!INTERNAL) {
16     separator = "&";
17     return "";
18   } else {
19     return x.getData(separator);
20   }
21 }
```



Javascript Hacking

```
1 /**
2  * @param {string} x
3  */
4 Request.prototype.getData = function (x) {
5   return
6   [ this.getUrl(), this.getParam(), this.getDate() ].join(x);
7 }
8
9 /**
10 * @return {string}
11 */
12 giffy.render = function () {
13   var x = new Respond();
14   var seperator;
15   if (!INTERNAL) {
16     return "";
17   } else {
18     seperator = "&";
19     return x.getData(seperator);
20   }
21 }
```



Compiling with Closure Compiler

WARNING - unreachable code

```
[this.getUrl(), this.getParam(), this.getDate()].join(x);
```



Javascript Hacking

```
1 /**
2  * @param {string} x
3  */
4 Request.prototype.getData = function (x) {
5   return
6   [ this.getUrl(), this.getParam(), this.getDate() ].join(x);
7 }
8
9 /**
10 * @return {string}
11 */
12 giffy.render = function () {
13   var x = new Respond();
14   var separator;
15   if (!INTERNAL) {
16     return "";
17   } else {
18     separator = "&";
19     return x.getData(separator);
20   }
21 }
```



Javascript Hacking

```
1 /**
2  * @param {string} x
3  */
4 Respond.prototype.getData = function (x) {
5   return [ this.getUrl(), this.getParam(), this.getDate() ].join(x);
6 }
7
8 /**
9  * @return {string}
10 */
11 giffy.render = function () {
12   var x = new Respond();
13   var separator;
14   if (!INTERNAL) {
15     return "";
16   } else {
17     separator = "&";
18     return x.getData(separator);
19   }
20 }
```



Why?

- Static type checking is useful sometimes.....
- Code Defensively (API)
- Better code, encourages readability documentations
- Errors now, instead error later.

Optimizations

Optimizations

- Smaller -> Faster
- Faster -> Better

Optimizations

- Smaller -> Faster
- Faster -> Better
- **Smaller -> More features!**

Example Compression: Closure's Date Picker Example

goog.ui.DatePicker

Default: ISO 8601

« 2010 »		« April »						
Mon	Tue	Wed	Thu	Fri	Sat	Sun		
13	29	30	31	1	2	3	4	
14	5	6	7	8	9	10	11	
15	12	13	14	15	16	17	18	
16	19	20	21	22	23	24	25	
17	26	27	28	29	30	1	2	
18	3	4	5	6	7	8	9	
Today						None		

2010-04-05

English (US)

« May »		« 2010 »					
Sun	Mon	Tue	Wed	Thu	Fri	Sat	
17	25	26	27	28	29	30	1
18	2	3	4	5	6	7	8
19	9	10	11	12	13	14	15
20	16	17	18	19	20	21	22
21	23	24	25	26	27	28	29
22	30	31	1	2	3	4	5
Today						None	

2010-05-12

Custom

- ShowFixedNumWeeks
- ShowOtherMonths
- ExtraWeekAtEnd
- ShowWeekNum
- ShowWeekdays
- AllowNone
- ShowToday
- UseNarrowWeekdayNames
- UseSimpleNavigationMenu

« 2006 »		« January »						
Mon	Tue	Wed	Thu	Fri	Sat	Sun		
52	26	27	28	29	30	31	1	
1	2	3	4	5	6	7	8	
2	9	10	11	12	13	14	15	
3	16	17	18	19	20	21	22	

German

« Mai »		« 2010 »					
Mo.	Di.	Mi.	Do.	Fr.	Sa.	So.	
17	26	27	28	29	30	1	2
18	3	4	5	6	7	8	9
19	10	11	12	13	14	15	16
20	17	18	19	20	21	22	23
21	24	25	26	27	28	29	30
22	31	1	2	3	4	5	6
Today						None	

2010-05-12

Malayalam

« 2010 »		« May »					
Mon	Tue	Wed	Thu	Fri	Sat	Sun	
17	25	26	27	28	29	30	1
18	2	3	4	5	6	7	8
19	9	10	11	12	13	14	15
20	16	17	18	19	20	21	22
21	23	24	25	26	27	28	29
22	30	31	1	2	3	4	5
Today						None	

File Sizes for Date Picker Example

	Original	Compression: Whitespace	Compression: Simple	Compression: Advanced
Uncompressed size	701 KB (0%)	351 KB (50%)	301 KB (58%)	33 KB (95%)
Gzip size	146 KB (0%)	61 KB (58%)	55 KB (62%)	12 KB (92%)
Time to load page		800 ms	700 ms	474 ms

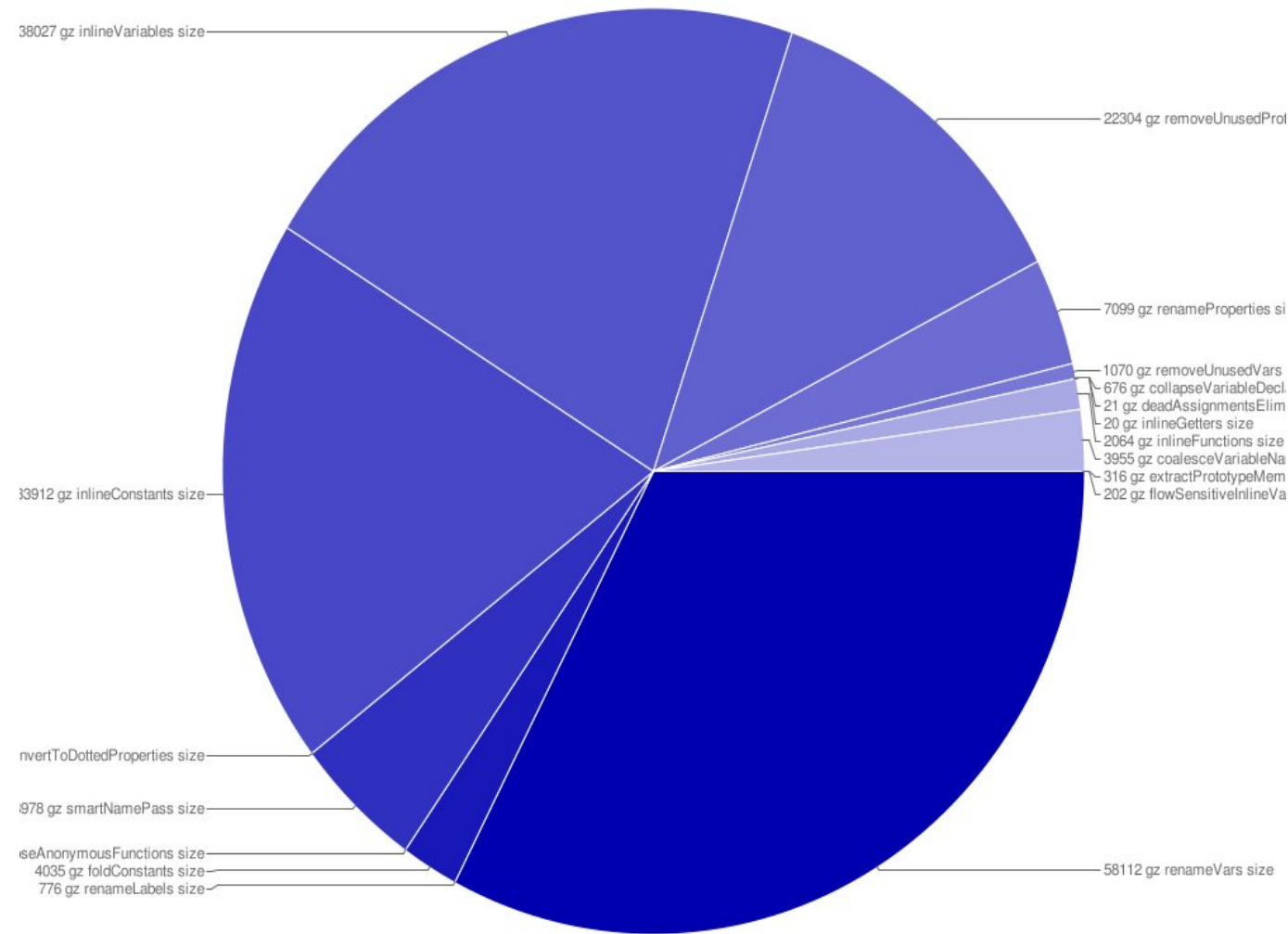
Optimizations

- Rename Variables
- Rename Properties
- Rename Labels

Lots of Optimizations

- Rename Variables
- Rename Properties
- Rename Labels
- Inline Functions
- Inline Variables
- Optimize Arguments
- Chain Calls
- Remove Dead Code
- Remove Dead Assignments
- Remove Unused Variables
- Remove Unused Properties
- Alias Strings
- Alias Keywords
- Collapse Properties
- Devirtualize Functions
- Side Effect Computations
- Rewrite Common Functions
- Coalesce Variable Names
- **MANY MORE!!!!**

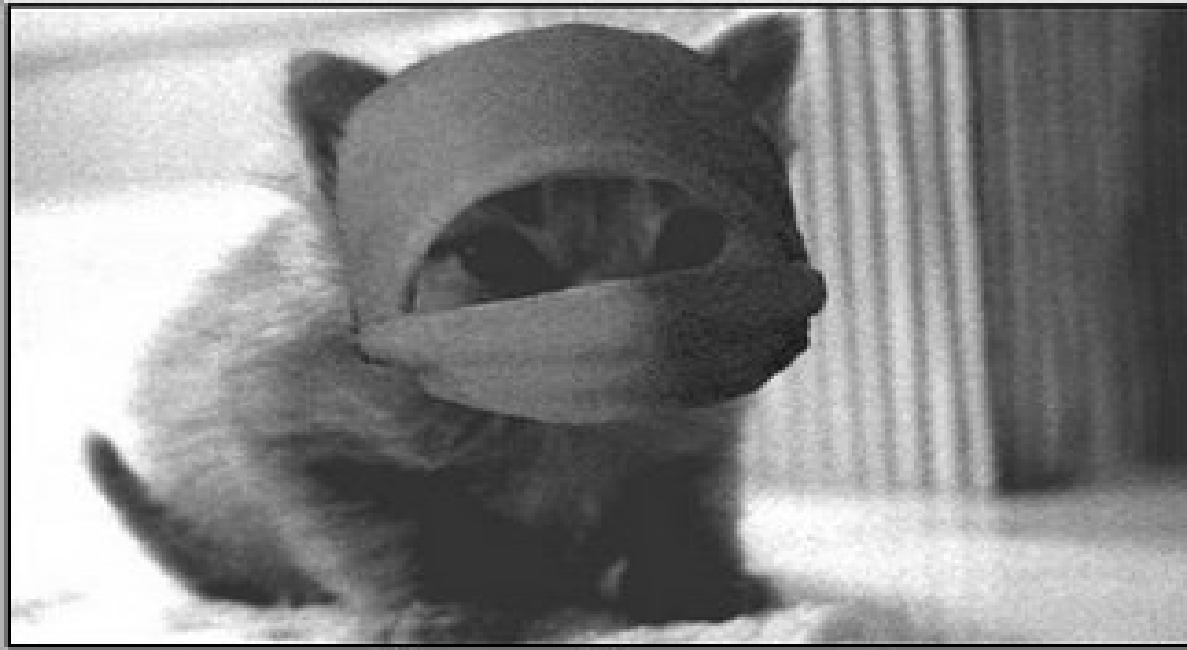
Lots of Optimizations



~30% Renaming
~30% Inlining
~15% Dead Code



Optimizations?



NINJA

Now available in kitten

- Javascript Ninja skillz!



What exactly does it do?

- "Make extensive use of" (**abuse**)
every Javascript operator / language semantics

What exactly does it do?

- "Make extensive use of" (**abuse**)
every Javascript operator / language semantics

```
a=typeof a==="object"?[c.expando]?a:c.extend(c.Event(f),a):  
c.Event(f);
```

- Programmer can focus only on their logic and readability of the code.

Examples of Optimizations: Constant Folding

```
if (x < (lineLength - (ellipsisLength * 2))) {
```

becomes

```
if (x < 74) {
```



Examples of Optimizations: Coalesce Variable Names

```
myVar = 3; otherVar = myVar + c; fVar = otherVar++;
```

becomes

```
a =3; a = a + c; a++;
```



Examples of Optimizations: Function Inlining

```
function fieldValue() {  
    return field + offset;  
}  
y = fieldValue();
```

becomes

```
y = field + offset;
```

Examples of Optimizations: Simple Common Subexpression

```
Foo.prototype.bar1 = ....  
Foo.prototype.bar2 = ....
```

becomes

```
t = Foo.prototype;  
t.bar1 = ...  
t.bar2 = ...
```

Examples of Optimizations: Collapse Properties

```
goog.ui.messages.foo = function() { ....  
goog.ui.messages.foo();
```

becomes

```
goog$ui$messages$foo = function () { ...  
goog$ui$messages$foo();
```

becomes (with Variable Renaming)

```
a = function() {.,...  
a();
```



What else is new?

1) disposeInternal() -> aaaa()
addDocument() -> bbbb()

What else is new?

1) disposeInternal() -> aaaa()
addDocument() -> bbbb()

2) disposeInternal() -> aaaa()
addDocument() -> aaaa()

What else is new?

- 1) disposeInternal() -> aaaa()
addDocument() -> bbbb()
- 2) disposeInternal() -> aaaa()
addDocument() -> aaaa()
- 3) disposeInternal() -> aaaa()
addDocument() -> baaa()

What else is new?

1) disposeInternal() -> aaaa()
addDocument() -> bbbb()

2) disposeInternal() -> aaaa()
addDocument() -> aaaa()

3) disposeInternal() -> aaaa()
addDocument() -> baaa()

NP-Complete Problem!



Basic Usage



Basic Usage

- Whitespace Mode
- Simple Mode
- Advance Mode (Some extra work)

Basic Usage

- Whitespace Mode
- Simple Mode
- Advance Mode (Some extra work)

```
window['displayNoteTitle'] = displayNoteTitle;
```

More Advanced Usage



Defining Constants

```
1 /**
2  * @param {string} x
3  */
4 Respond.prototype.getData = function (x) {
5   return [ this.getUrl(), this.getParam(), this.getDate() ].join(x);
6 }
7
8 /**
9  * @return {string}
10 */
11 giffy.render = function () {
12   var x = new Respond();
13   var seperator;
14   if (!INTERNAL) {
15     return "";
16   } else {
17     seperator = "&";
18     return x.getData(seperator);
19   }
20 }
```



Defining Constants

```
java -jar compiler.jar \  
  --js giffy-uncompiled.js --js a.js \  
  --js_output_file giffy-compiled.js \  
  --define INTERNAL=true \  
  --jscomp_error checkTypes \  
  --compilation_level ADVANCED_OPTIMIZATIONS
```

Defining Constants

```
1 /**
2  * @param {string} x
3  */
4 Respond.prototype.getData = function (x) {
5   return [ this.getUrl(), this.getParam(), this.getDate() ].join(x);
6 }
7
8 /**
9  * @return {string}
10 */
11 giffy.render = function () {
12   var x = new Respond();
13   var seperator;
14   if (!INTERNAL) {
15     return "";
16   } else {
17     seperator = "&";
18     return x.getData(seperator);
19   }
20 }
```



Defining Constants

```
7
8 /**
9  * @return {string}
10 */
11 giffy.render = function () {

15   return "";

20 }
```

Type Inferencing

```
/** @constructor */  
function Apple() {}
```

```
/** @param {Apple!} a */  
function store(a) {}
```

Type Inferencing

```
/** @constructor */  
function Apple() {}
```

```
/** @param {Apple!} a */  
function store(a) {}
```

```
var x = getRandomFruit();
```

```
if (x instanceof Apple) {  
  var y = x;  
  ....  
  store(y);  
}
```

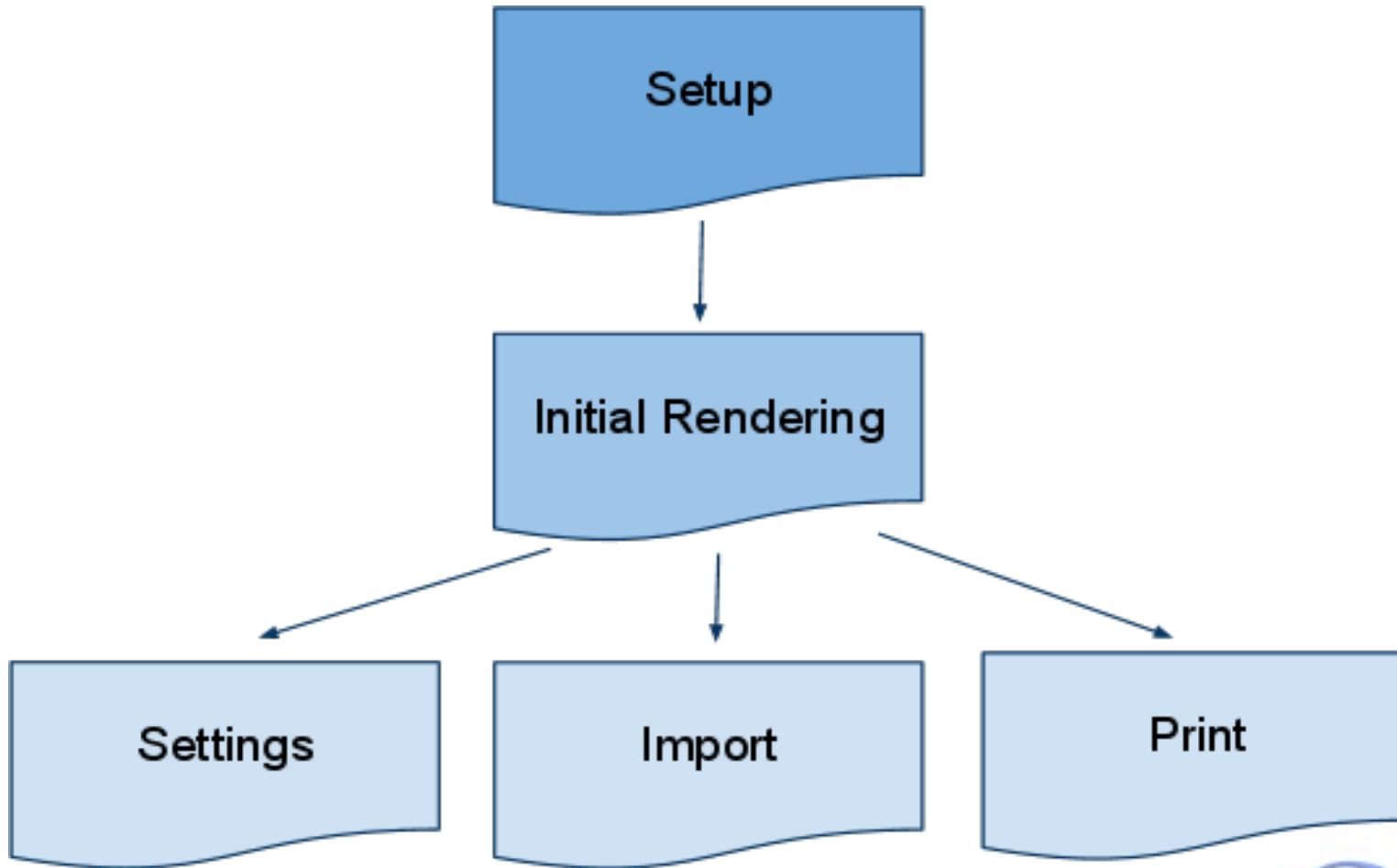
Modules?

Modules System!



Cross Module Code Motion

Modules



Cross Module Code Motion

```
// Init Mod
```

```
function Graph() { ... }  
Graph.prototype.display = function() { ... };  
Graph.prototype.resize = function() { ... };  
var g = new Graph();
```

```
// Render Mod
```

```
g.display();
```

```
// Resize Mod
```

```
g.resize();
```



Cross Module Code Motion

```
// Init Mod
```

```
function Graph() { ... }
```

```
Graph.prototype.display = function() { ... };
```

```
Graph.prototype.resize = function() { ... };
```

```
var g = new Graph();
```

```
// Render Mod
```

```
g.display();
```

```
// Resize Mod
```

```
g.resize();
```



Cross Module Code Motion

```
// Init Mod
```

```
function Graph() { ... }
```

```
var g = new Graph();
```

```
// Render Mod
```

```
Graph.prototype.display = function() { ... };
```

```
g.display();
```

```
// Resize Mod
```

```
Graph.prototype.resize = function() { ... };
```

```
g.resize();
```



Module Specialization

Even more aggressive initial module optimization!



Module Specialization

```
// Module 1
```

```
function x () { return 1 };  
alert(x() + x());
```

```
...
```

```
// Module 2 (depends on 1)
```

```
alert(x);  
x = function() { return 2 };
```

```
// Module 3 (depends on 1)
```

```
alert(x);
```

Module Specialization

```
// Module 1
```

```
function x () { return 1 };
```

```
alert(x() + x());
```

```
...
```

```
// Module 2 (depends on 1)
```

```
alert(x);
```

```
x = function() { return 2 };
```

```
// Module 3 (depends on 1)
```

```
alert(x);
```

Module Specialization

```
// Module 1
```

```
alert(2);
```

```
...
```

```
// Module 2 (depends on 1)
```

```
function x() { return 1 };
```

```
alert(x);
```

```
x = function() { return 2 };
```

```
// Module 3 (depends on 1)
```

```
function x() { return 1 };
```

```
alert(x);
```



Module Specialization

End Result

- Every one byte saved in initial module
- Increases every later modules by one byte.

Future?

- JQuery Integration
- Analysis tool
- Optimizations, size reduction
- More delay loading / module features
- EcmaScript 5 support

Summary

- Find bugs, Find errors, Static Types
- Best Javascript compression tool without any additional work
- Can reduce size greatly if you put in some work
- Great module system for delay loading, optimization algorithm to further decrease your initial download size

Questions?

- Alan Leung (acleung [at] gmail.com / google.com)
- Twitter @acleung
- Website, Docs, Mailing Lists, Bug reports:
<http://code.google.com/p/closure-compiler/>
- Online version
<http://closure-compiler.appspot.com/>
- These slides can be found at http://acleung.com/twitter_talk.pdf