



CS472 WAP

2 Pass Compiler

Scope, Closures, and Encapsulation

Except where otherwise noted, the contents of this document are Copyright 2012 Marty Stepp, Jessica Miller, Victoria Kirst and Roy McElmurry IV. All rights reserved. Any redistribution, reproduction, transmission, or storage of part or all of the contents in any form is prohibited without the author's expressed written permission. Slides have been modified for Maharishi University of Management Computer Science course CS472 in accordance with instructors agreement with authors.

Maharishi International University

Fairfield, Iowa © 2020



All rights reserved. No part of this slide presentation may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying or recording, or by any information storage and retrieval system, without permission in writing from Maharishi International University.

Main Point Preview

- JavaScript has a 2-pass compiler that hoists all function and variable declarations. These declarations are visible anywhere in the current function scope regardless of where they are declared. Variables have value 'undefined' until the execution pass and an assignment is made.
- *Science of Consciousness: The first pass sets up the proper conditions for the successful execution of the second pass. Similarly, when we set up the proper conditions for transcending then all we have to do is let go and nature will ensure that the experience is successful.*

Code Execution and Hoisting: 2 phase compilation

- When code executes, JS engine first creates global environment objects and “this” object and looks for function and variable declarations
- In **first phase**, JS engine looks through all global code for function declarations and global variables (hoisting)
 - functions: saves entire function definition
 - variables: saves only variable name and value of ‘undefined’
 - ‘hoists’ variable and function declarations
 - No variable initialization or function expressions are hoisted
- In **second phase**, JS engine
 - executes code line-by-line
 - for every function call adds new context and to execution context stack





Variable Declarations Are Hoisted

- JavaScript hoists all variable declarations
 - moves them to the beginning of their direct scopes (function).

```
function f(){  
  console.log(bar); //undefined  
  var bar = "abc";  
  console.log(bar); //abc  
}
```

- JavaScript executes f() as if its code were:

```
function f(){  
  var bar;  
  console.log(bar); //undefined  
  bar = "abc";  
  console.log(bar); //abc  
}
```

Function Declaration Hoisting

- Hoisting: moving to the beginning of a scope.
- Function declarations are hoisted completely
 - allows calling a function before it is declared.

```
foo();  
function foo(){ }
```

- JavaScript executes the code as if it looked like this:

```
function foo(){ }  
foo();
```

Function expressions are not hoisted

- Function expressions are not hoisted, so cannot use function expression functions before they are defined.

```
foo(); //TypeError: undefined is not a function
```

```
var foo = function (){  
    ...  
};
```

- JS Engine executes the code as:

```
var foo;  
foo(); //TypeError: undefined is not a function  
var foo = function (){  
    ...  
};
```




Hoisting Example

```
var a = 5;  
function b() {  
  console.log("function is called!");  
}  
console.log("a: " + a); //5  
b(); // function is called!
```

- Notice what will happen when we move lines:

```
console.log("a: " + a); //undefined  
b(); // function is called!
```

```
var a = 5;  
function b() {  
  console.log("function is called!");  
}
```

- What will happen if remove the variable a declaration line?

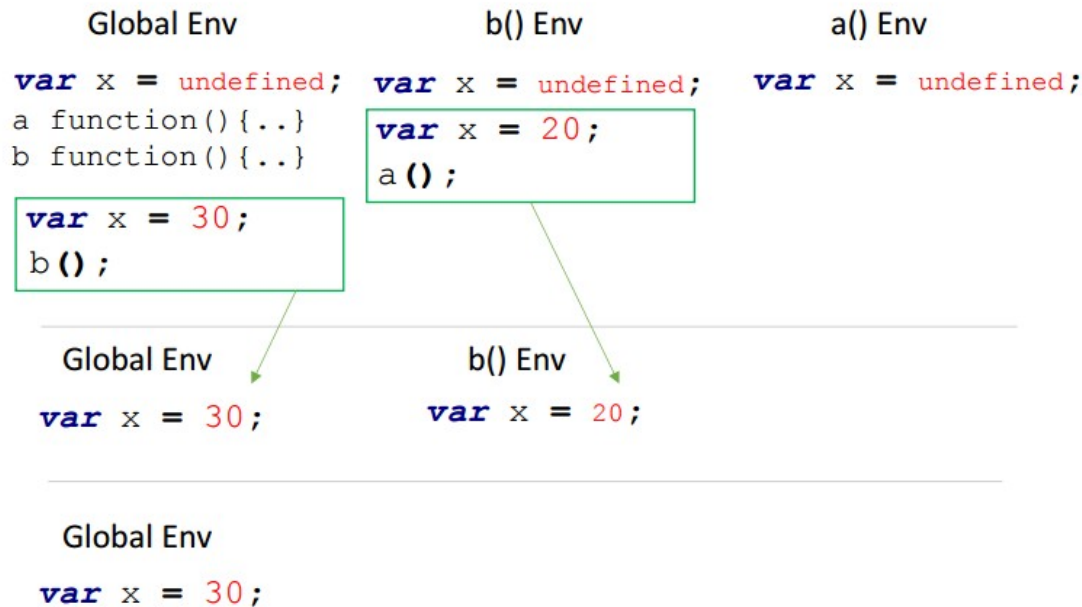


Execution context & stack example

```
function a() {
  var x;
}
```

```
function b() {
  var x = 20;
  a();
}
```

```
var x = 30;
b();
console.log(x);
```



Main Point

- JavaScript has a 2-pass compiler that hoists all function and variable declarations. These declarations are visible anywhere in the current function scope regardless of where they are declared. Variables have value 'undefined' until the execution pass and an assignment is made.
- *Science of Consciousness: The first pass sets up the proper conditions for the successful execution of the second pass. Similarly, when we set up the proper conditions for transcending then all we have to do is let go and nature will ensure that the experience is successful.*

