3416 JavaScript and AngularJS

Thank you!

Thank you for choosing the University of Toronto School of Continuing Studies

Our Courses & Programs

The School offers more than 600 courses in 80 certificates, both classroom-based and online, covering a vast range of interests and specializations:

- Business & Professional Studies
- English Language Program
- Arts & Science
- Languages & Translation
- Creative Writing

Follow us on social

Join the conversation with us online:

facebook.com/UofTLearnMore

@UofTLearnMore

linkedin.com/company/university-of-toronto-school-of-continuing-studies

Module 2

INTRODUCTION TO JAVASCRIPT

Module 1: Learning Outcomes

- Introduction to JavaScript, JavaScript Scope, Scope Chain, Primitive Type, Object Type, String Concatenation, Equality, Default Values, Creating Objects using new Object and Object literal notation.

How to Declare JavaScript

- Three ways to declare javascript:

  <script type="text/javascript"></script>

- Create Example1.html

  <script ></script>

- Create Example2.html

  <script src="example3.js"></script>

- Create Example3.html

How to Declare JavaScript

- Declare javascript inside body.
- Create Example4.html

Defining Variables

- var message = “Hello Albert”;
- Variable definition should always start with ‘var’.
- No types are declared.
  - Javascript is dynamically typed language.
  - Javascript engine figure out the type of the variable at run-time.
  - Same variable can hold different types during the life of the execution. Variable can start off as a string and change to a number and than to a string.

Defining Functions

- function a() {..}
- The way to define a function is the keyword “function” followed by function name, followed by parentheses and than curly braces.

Another way to define a function

- var a = function () {..}
- Created a variable and set it equal to a function. No name is defined after function keyword.
- Value of function is assigned, NOT the returned result!

Way to Invoke a function

- a();
- Take the name of the function and putting parentheses afterward.
- Execution of a function is the same as invokes the function.

How To Define Arguments for a function

- Argument are defined without the keyword ‘var’.
- function compare(x,y)

```
{

  return x > y;
}
```

If you want the function to return a value type return with a value.

If return keyword doesn’t have a value you’re telling the javascript engine to terminate the function and exit out of it without returning anything.

See Example5.html

Scope

- Global Scope vs Function(lexical) scope.
- Variables and functions defined in the global scope are available everywhere. Other functions defined in the global scope can get access to these global defined variables.
- Variables and functions defined in lexical scope are available only within this function.

Scope Chain

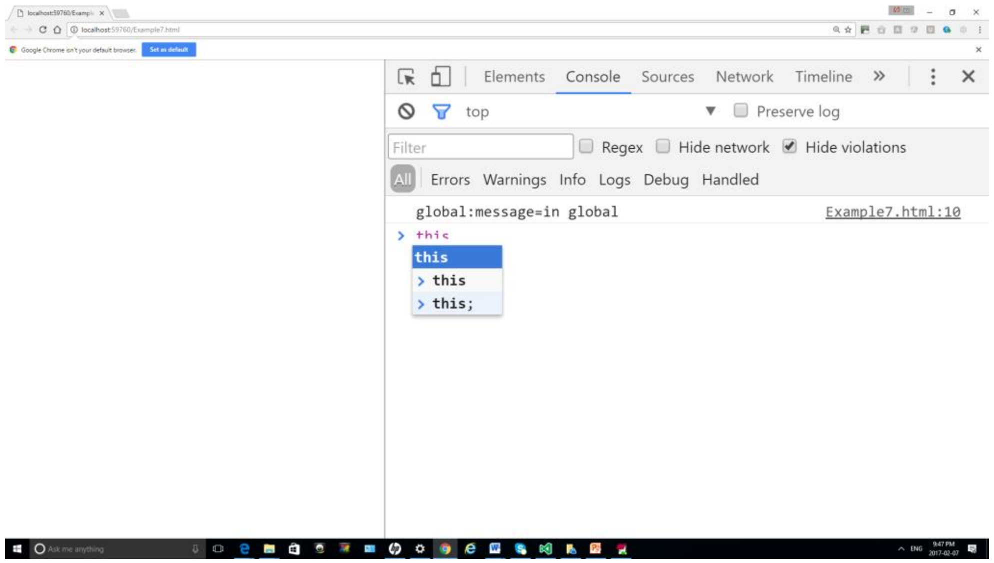
- Everything is executed in a Execution Context
- Function invocation creates a new Execution Context.
- Each Execution Context has:
  - Its own Variable Environment
  - Reference to its Outer Environment.
- Global scope does not have an Outer Environment as it's the most outer there is.

Scope Chain

- Reference(not defined) variable will be searched for in its current scope first. If not found, the Outer Reference will be searched.
- It not found, the Outer Reference’s Outer Reference will be searched, etc.
- This will keep going until the Global scope.
- If not found in Global scope, the variable is undefined.
- Create Example6.html

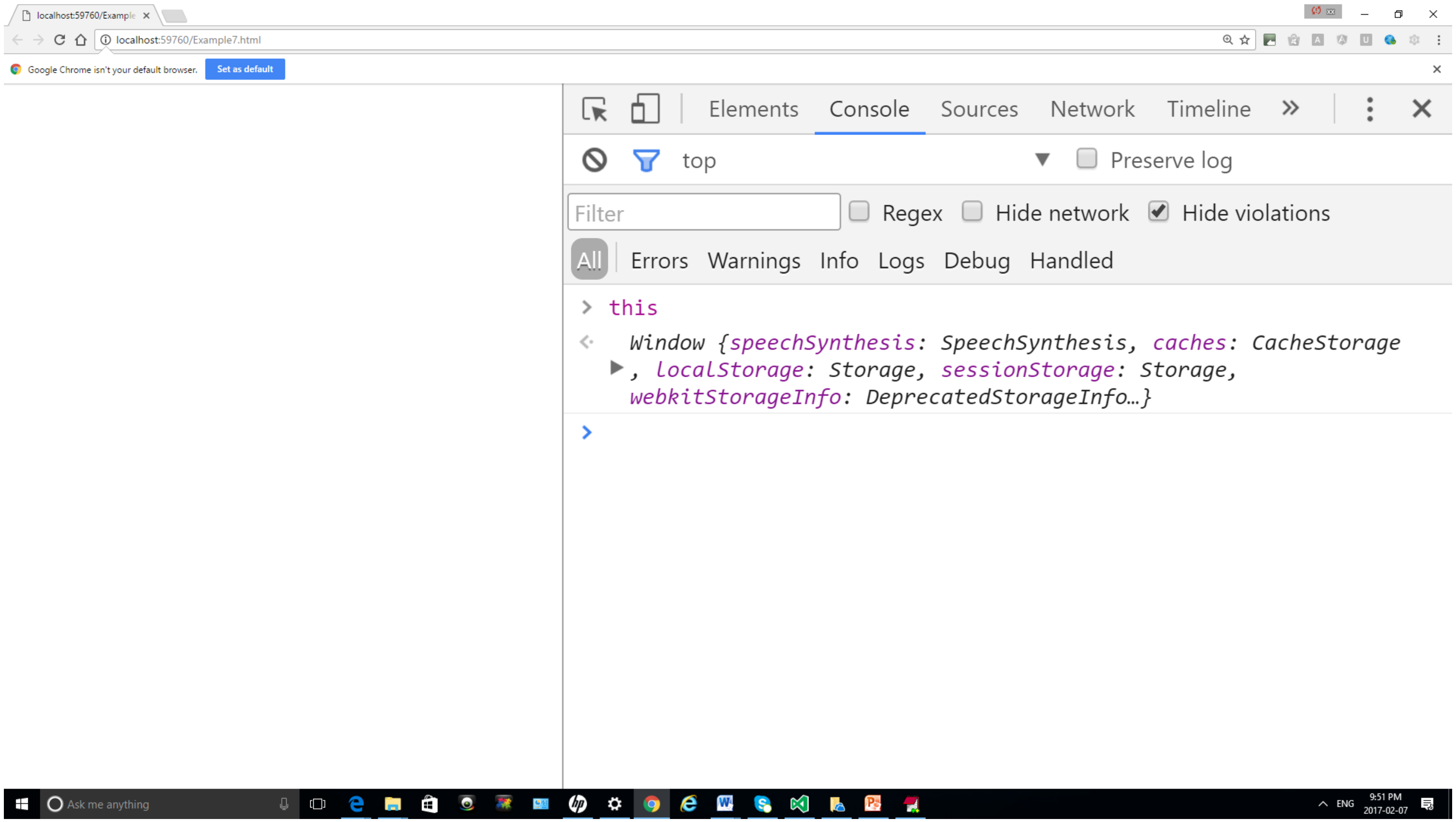
Scope Chain

- Create Example7.html



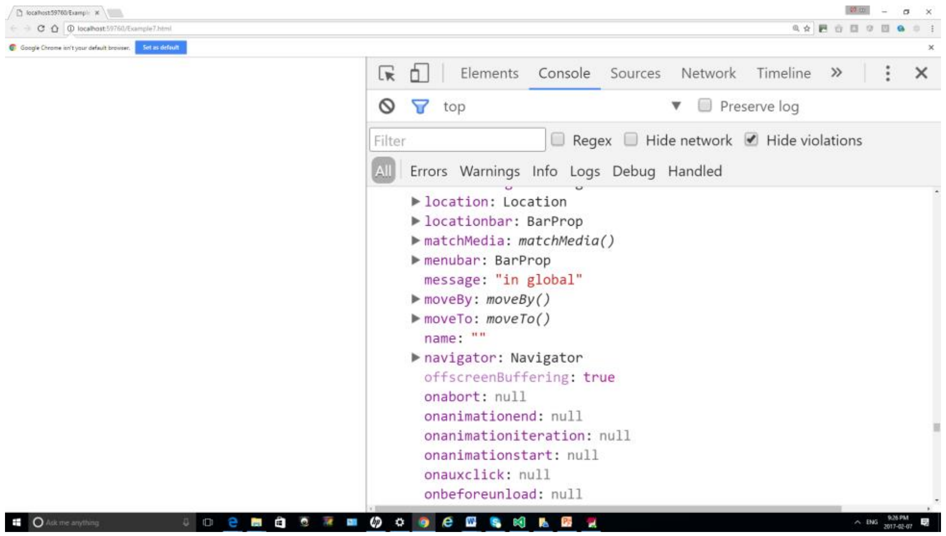
```
global:message=in global                Example7.html:10
> this
  this
  > this
  > this;
```

Scope Chain



```
> this
<· ▶ Window {speechSynthesis: SpeechSynthesis, caches: CacheStorage
  , localStorage: Storage, sessionStorage: Storage,
  webkitStorageInfo: DeprecatedStorageInfo…}
>
```

Scope Chain



```
▶ location: Location
▶ locationbar: BarProp
▶ matchMedia: matchMedia()
▶ menubar: BarProp
  message: "in global"
▶ moveBy: moveBy()
▶ moveTo: moveTo()
  name: ""
▶ navigator: Navigator
  offscreenBuffering: true
  onabort: null
  onanimationend: null
  onanimationiteration: null
  onanimationstart: null
  onauxclick: null
  onbeforeunload: null
```

Scope Chain

- Create Example8.html

JavaScript Built in Types

- A type is a particular data structure.
- Each language defines some built-in types.
- Built-in types can be used to build other data structures.
- Javascript has 6 built-in types: 5 primitive and 1 Object type.

Object Type

- Object is a collection of name/value pairs

```
var Employee = {
firstName: “Albert”,
lastName: “Lam”,
Social: {
                linkedin:”albertLam”,
                twitter:”albertLam”,
                facebook:”albertLam”
            }
}
```

name

value

Object Type

- Create Example9.html

Primitive Types

- Primitive type represents a single, immutable value
- Single value means it is not an object.
- Object is a collection of name value pairs.
- Immutable means once it’s set, it can’t be changed.
  - Value becomes read-only.
  - You can create another value based on an existing one
  - But the memory space for the first value is not changed instead a new memory space is create for the new value.

Primitive Type: Boolean

- Boolean can only have 2 values: true or false
- True or false are reserved key words in the javascript language.
- Create Example10.html
- Create Example10b.html
- Create Example10c.html

Primitive Type: Undefined

- Undefined signifies that no value has even been set on this particular variable of this type(a variable has been defined but not assigned a value).
- Can only have one value: undefined.
- This is a reserved key word.
- You can set a variable to undefined, but you should NEVER do it.
  - Its meaning is that it's never been defined, so defining it to undefined is counter to its core meaning.
  - Create Example11.html and Example12.html

Undefined Vs Not Defined

- Undefined means it has been defined but not being set or
- Undefined means variable memory has been allocated but no value has ever been explicitly set yet.
- Not defined means the variable has never being defined or declared.
- Create Example12b.html

Primitive Type:Null

- Null signifies lack of value.
- Can only have one value: null
- It’s ok to explicitly set a variable to null
- Create Example13.html

Primitive Type:Number

- Number is the only numeric type in Javascript
- Always represented under the hood as double-precision 64-bit floating point.
- JS does not have an integer type.
  - Integers are a subset of doubles instead of a separate data type.
  - Create Example14.html

String

- String is sequence of characters used to represent text.
- Use either single or double quotes, i.e., ‘text’ or “text”.
- Create Example15.html

String Concatenation

- Create Example16.html

Regular Math Operators:+,-,*,/

- Create Example17.html

Equality

- Create Example18.html
- Strict Equality
  - – Create Example19.html

Curly Brace on the same or next line

- Create Example20.html

For Loop

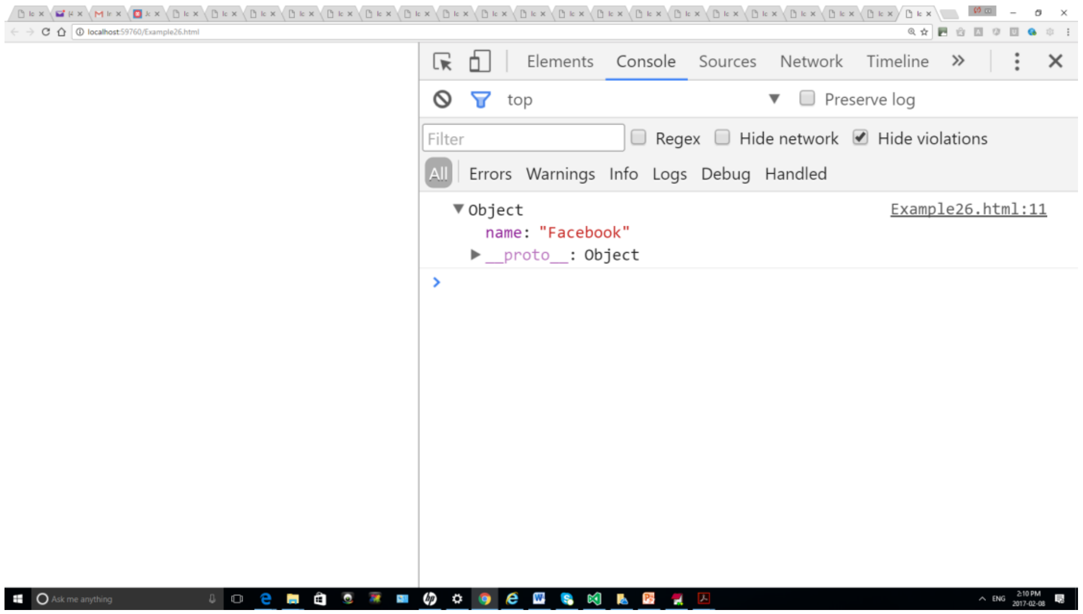
- Create Example21.html

Handling Default Values

- Create Example22.html, Example23.html, Example24.html, Example25.html

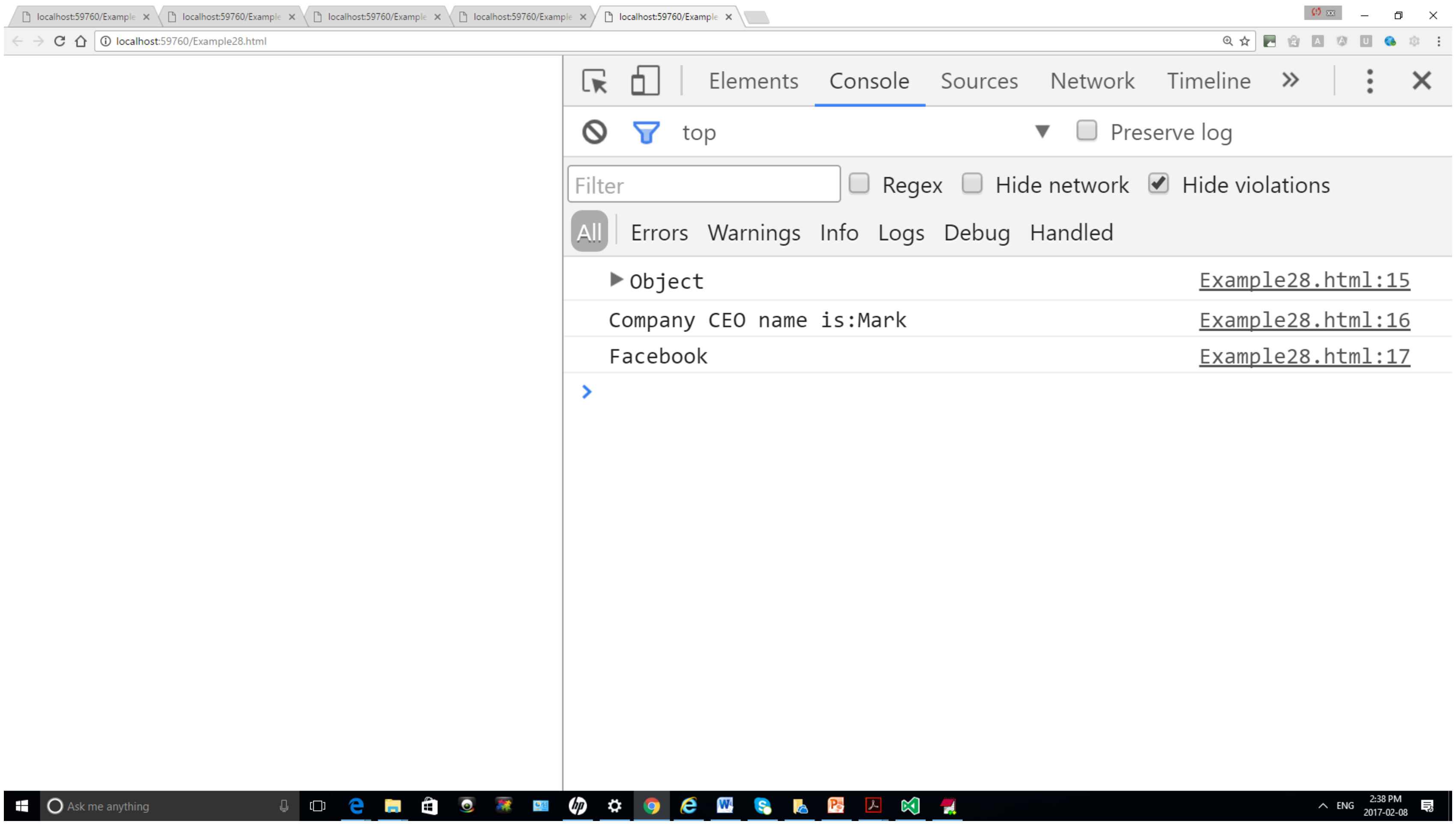
Creating objects using new Object

- Create Example26.html, Example27.html, Example28.html



```
▼Object                                          Example26.html:11
   name: "Facebook"
 ▶ __proto__: Object
>
```

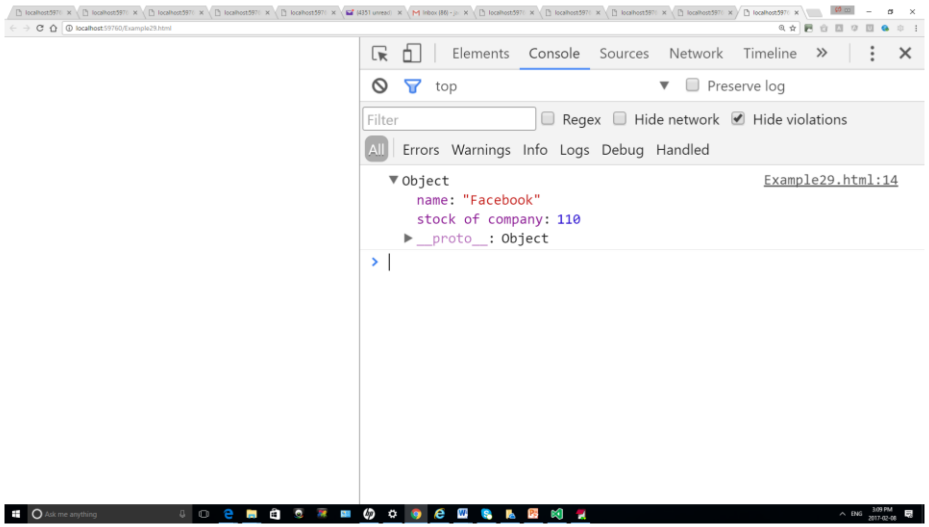
Example28.html



```
▶ Object                                   Example28.html:15
Company CEO name is:Mark                   Example28.html:16
Facebook                                   Example28.html:17
```

Dot notation vs [] notation

- Create Example29.html and Example30.html,



```
▼Object                                          Example29.html:14
   name: "Facebook"
   stock of company: 110
 ▶__proto__: Object
```

Object literal notation

- Two ways to create objects one is using “new” and the other way is object literal.
- Example
- var facebook = {

```
    firstName: "Facebook",
    ceo: {          //beginning object literal
        firstName: "Mark",
        favColor: "blue"
    },              //end object literal
    "stock of company": 110
};
```

- Create Example31.html

Functions

- Functions are First-Class Data Types
- Functions are Objects
- Create Example32.html