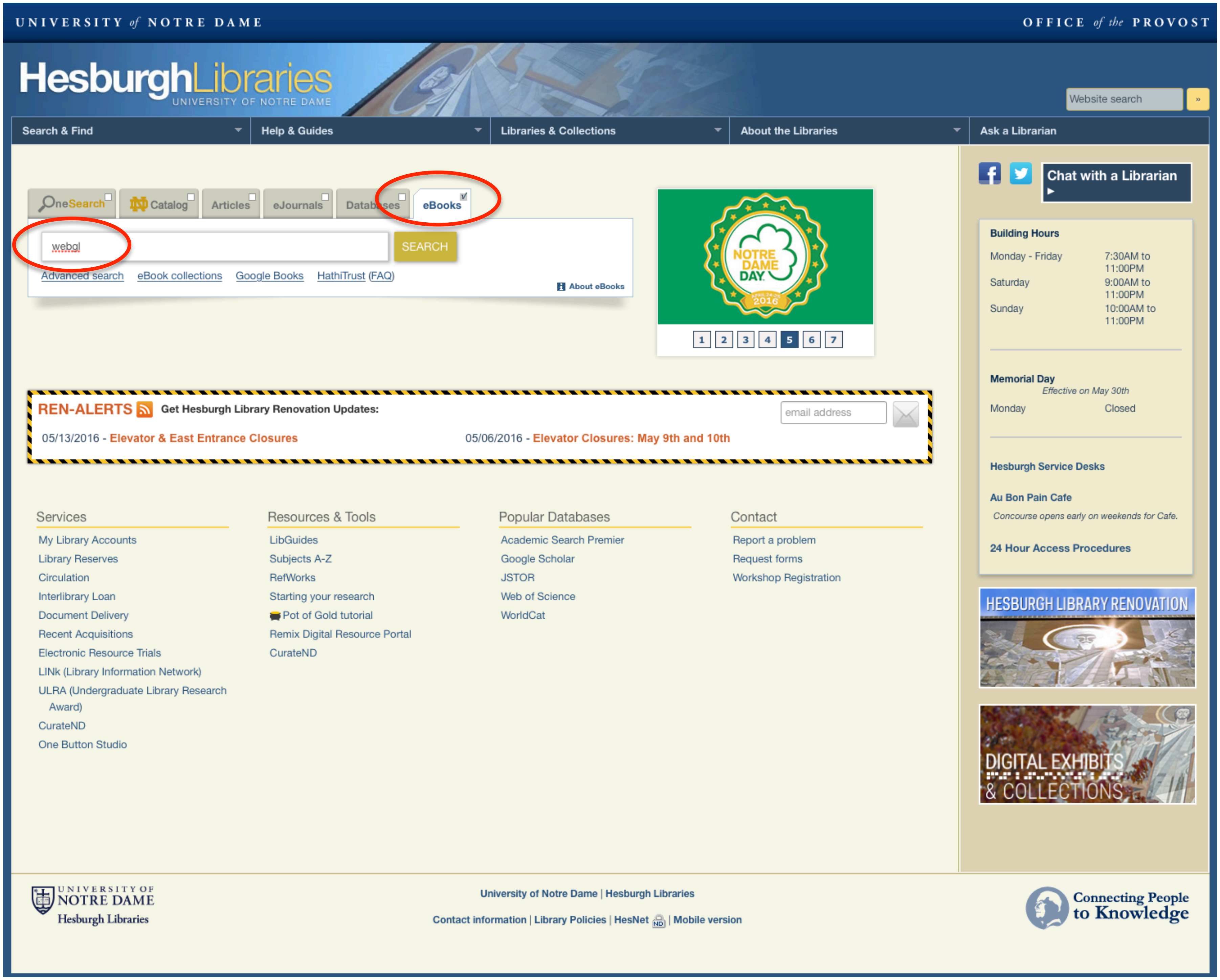
Online e-Books via ND Hesburgh Libraries


UNIVERSITY of NOTRE DAME

OFFICE of the PROVOST

HesburghLibraries UNIVERSITY OF NOTRE DAME

Website search »

Search & Find | Help & Guides | Libraries & Collections | About the Libraries | Ask a Librarian

OneSearch | Catalog | Articles | eJournals | Databases | eBooks

webgl SEARCH

Advanced search eBook collections Google Books HathiTrust (FAQ)

About eBooks

1 2 3 4 5 6 7

Chat with a Librarian ►

Building Hours

Monday - Friday	7:30AM to 11:00PM
Saturday	9:00AM to 11:00PM
Sunday	10:00AM to 11:00PM

Memorial Day

Effective on May 30th

Monday	Closed

Hesburgh Service Desks

Au Bon Pain Cafe

Concourse opens early on weekends for Cafe.

24 Hour Access Procedures

HESBURGH LIBRARY RENOVATION

DIGITAL EXHIBITS & COLLECTIONS

REN-ALERTS Get Hesburgh Library Renovation Updates: email address

05/13/2016 - Elevator & East Entrance Closures

05/06/2016 - Elevator Closures: May 9th and 10th

Services
- My Library Accounts
- Library Reserves
- Circulation
- Interlibrary Loan
- Document Delivery
- Recent Acquisitions
- Electronic Resource Trials
- LINk (Library Information Network)
- ULRA (Undergraduate Library Research Award)
- CurateND
- One Button Studio

Resources & Tools
- LibGuides
- Subjects A-Z
- RefWorks
- Starting your research
- Pot of Gold tutorial
- Remix Digital Resource Portal
- CurateND

Popular Databases
- Academic Search Premier
- Google Scholar
- JSTOR
- Web of Science
- WorldCat

Contact
- Report a problem
- Request forms
- Workshop Registration

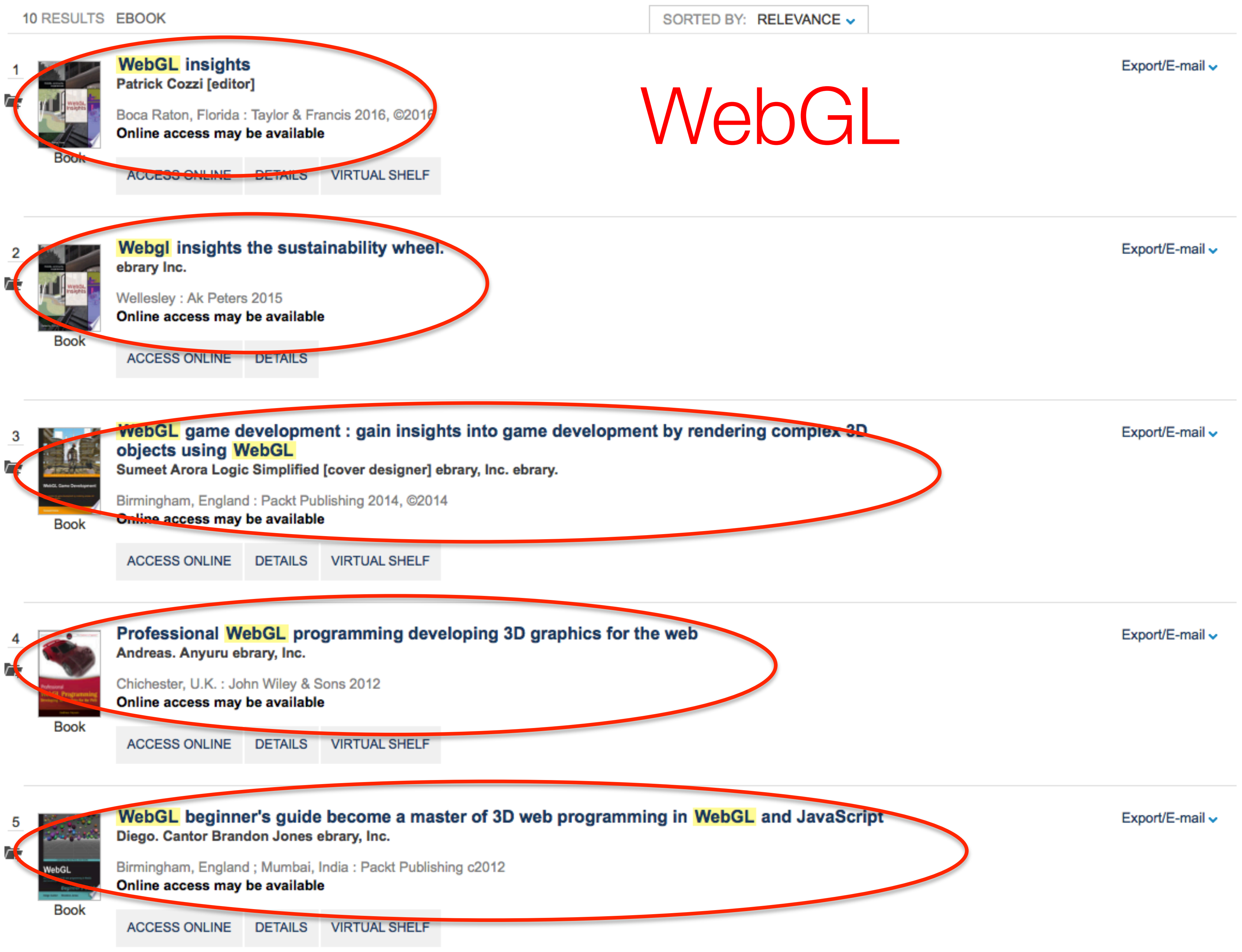
UNIVERSITY OF NOTRE DAME Hesburgh Libraries

University of Notre Dame | Hesburgh Libraries

Contact information | Library Policies | HesNet | Mobile version

Connecting People to Knowledge

WebGL



10 RESULTS EBOOK

SORTED BY: RELEVANCE

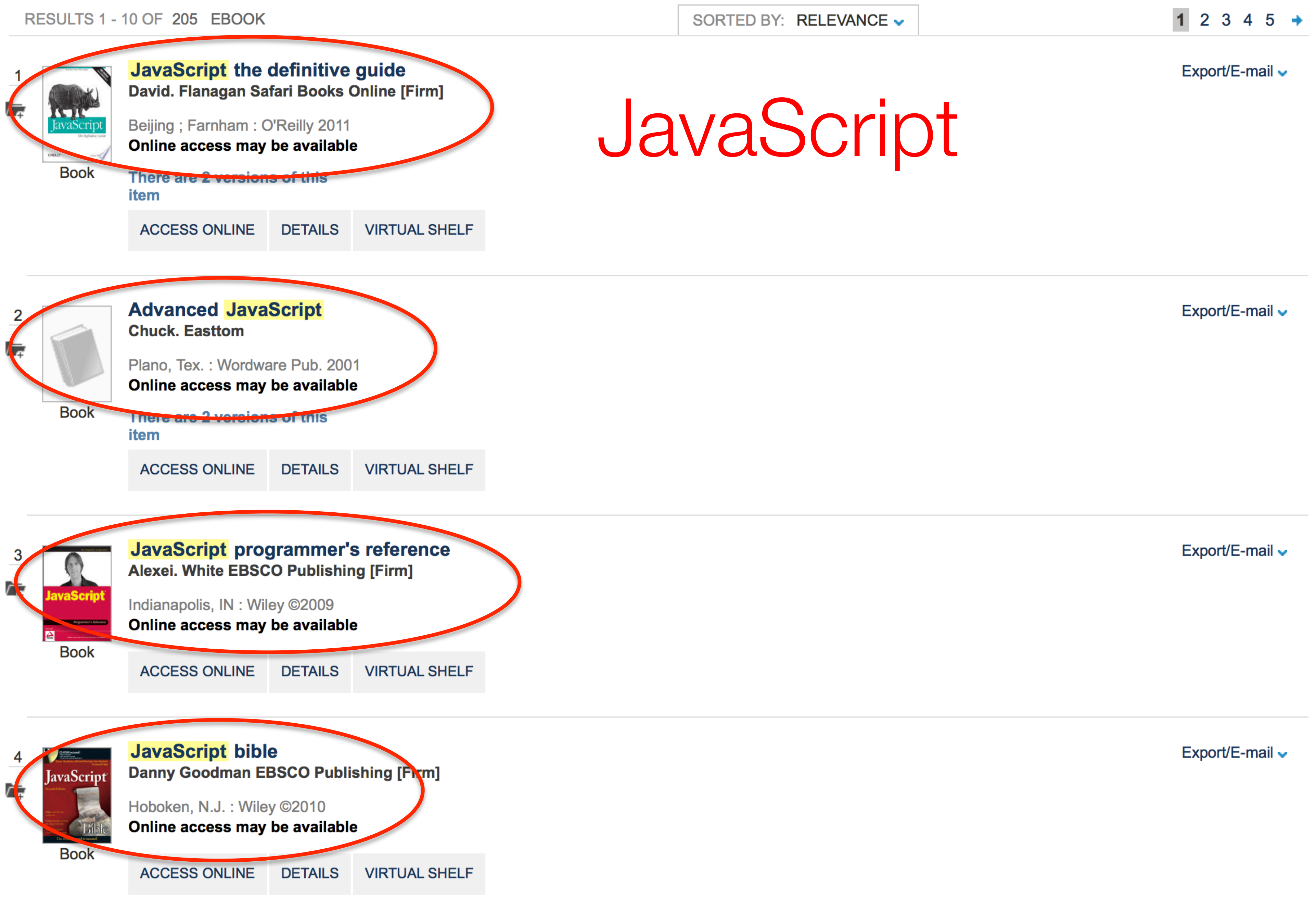
1 **WebGL insights**
Patrick Cozzi [editor]
Boca Raton, Florida : Taylor & Francis 2016, ©2016
Online access may be available
Book
ACCESS ONLINE | DETAILS | VIRTUAL SHELF
Export/E-mail

2 **Webgl insights the sustainability wheel.**
ebrary Inc.
Wellesley : Ak Peters 2015
Online access may be available
Book
ACCESS ONLINE | DETAILS
Export/E-mail

3 **WebGL game development : gain insights into game development by rendering complex 3D objects using WebGL**
Sumeet Arora Logic Simplified [cover designer] ebrary, Inc. ebrary.
Birmingham, England : Packt Publishing 2014, ©2014
Online access may be available
Book
ACCESS ONLINE | DETAILS | VIRTUAL SHELF
Export/E-mail

4 **Professional WebGL programming developing 3D graphics for the web**
Andreas. Anyuru ebrary, Inc.
Chichester, U.K. : John Wiley & Sons 2012
Online access may be available
Book
ACCESS ONLINE | DETAILS | VIRTUAL SHELF
Export/E-mail

5 **WebGL beginner's guide become a master of 3D web programming in WebGL and JavaScript**
Diego. Cantor Brandon Jones ebrary, Inc.
Birmingham, England ; Mumbai, India : Packt Publishing c2012
Online access may be available
Book
ACCESS ONLINE | DETAILS | VIRTUAL SHELF
Export/E-mail



RESULTS 1 - 10 OF 205 EBOOK

SORTED BY: RELEVANCE

1 2 3 4 5

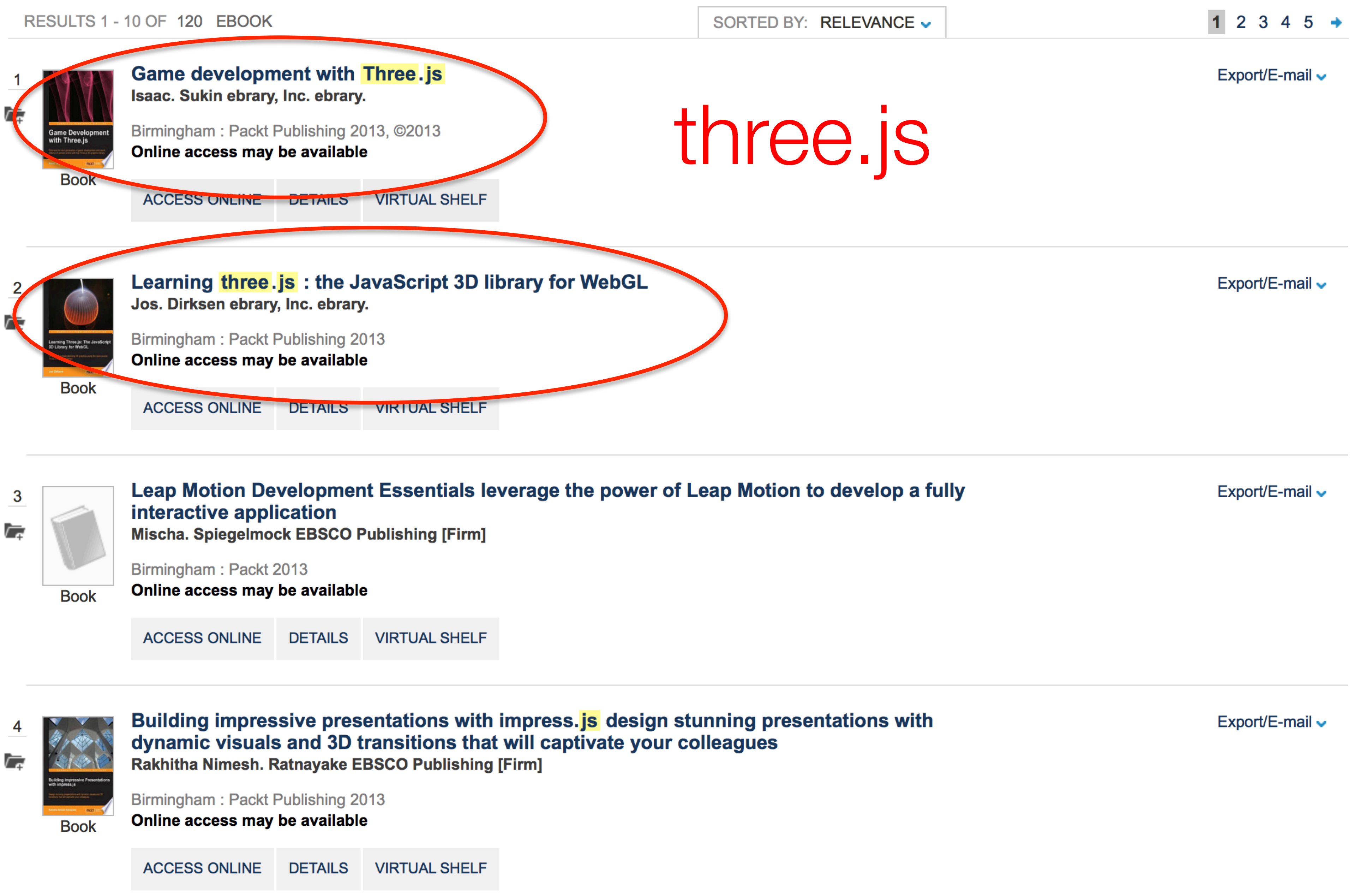
JavaScript

1 **JavaScript the definitive guide**
David. Flanagan Safari Books Online [Firm]
Beijing ; Farnham : O'Reilly 2011
Online access may be available
There are 2 versions of this item
Book
ACCESS ONLINE | DETAILS | VIRTUAL SHELF
Export/E-mail

2 **Advanced JavaScript**
Chuck. Easttom
Plano, Tex. : Wordware Pub. 2001
Online access may be available
There are 2 versions of this item
Book
ACCESS ONLINE | DETAILS | VIRTUAL SHELF
Export/E-mail

3 **JavaScript programmer's reference**
Alexei. White EBSCO Publishing [Firm]
Indianapolis, IN : Wiley ©2009
Online access may be available
Book
ACCESS ONLINE | DETAILS | VIRTUAL SHELF
Export/E-mail

4 **JavaScript bible**
Danny Goodman EBSCO Publishing [Firm]
Hoboken, N.J. : Wiley ©2010
Online access may be available
Book
ACCESS ONLINE | DETAILS | VIRTUAL SHELF
Export/E-mail



RESULTS 1 - 10 OF 120 EBOOK

SORTED BY: RELEVANCE

1 2 3 4 5

three.js

1 **Game development with Three.js**
Isaac. Sukin ebrary, Inc. ebrary.
Birmingham : Packt Publishing 2013, ©2013
Online access may be available
Book
ACCESS ONLINE DETAILS VIRTUAL SHELF
Export/E-mail

2 **Learning three.js : the JavaScript 3D library for WebGL**
Jos. Dirksen ebrary, Inc. ebrary.
Birmingham : Packt Publishing 2013
Online access may be available
Book
ACCESS ONLINE DETAILS VIRTUAL SHELF
Export/E-mail

3 **Leap Motion Development Essentials leverage the power of Leap Motion to develop a fully interactive application**
Mischa. Spiegelmock EBSCO Publishing [Firm]
Birmingham : Packt 2013
Online access may be available
Book
ACCESS ONLINE DETAILS VIRTUAL SHELF
Export/E-mail

4 **Building impressive presentations with impress.js design stunning presentations with dynamic visuals and 3D transitions that will captivate your colleagues**
Rakhitha Nimesh. Ratnayake EBSCO Publishing [Firm]
Birmingham : Packt Publishing 2013
Online access may be available
Book
ACCESS ONLINE DETAILS VIRTUAL SHELF
Export/E-mail

WebGL Examples

http://www.awwwards.com/22-experimental-webgl-demo-examples.html

Please bring your laptop to class, we will do in-class coding exercises to make sure that everyone knows how to write, run, and debug WebGL programs!

WebGL is based on HTML5, GLSL and JavaScript. Let us start with a brief tutorial of JavaScript.

Learning Objectives

- Students completing this lecture will be able to
  - Describe the major characteristics of JavaScript language
  - Explain the need and benefit for executing JavaScript in “strict mode”
  - Write simple JavaScript code (e.g., variable declaration, input, output) with reference to this set of slides
  - Set up browser environment (Firefox + Firebug)

JavaScript Tutorial

JavaScript (JS)

- An interpreted language with a C like syntax
- A browser scripting language (the language of the Web)
- All browsers will execute JS code
- Approachable for the beginner
- You just need a simple text-editor and a browser to get started

Getting Started

```
<html>
  <head>
    <title>Learning Javascript</title>
  </head>
  <body>
    <p>Hello World!
  </body>
</html>
```

In-line JS

```
<script type='text/javascript'>
// Your script goes here.
</script>
```

- Want the script blocks to appear where you want their output to be
  - If I wanted to say “Hello World!” I would want my script block to appear in the `<body>` area of my web page and not in the `<head>` section
- Good practice says that you should place your scripts at the very bottom of your HTML
  - Each time the browser encounters a `<script>` tag it has to pause, compile the script, execute the script, then continue on generating the page

External JS

```
<script type='text/javascript'
src='common.js'></script>
```

- Everything that would ordinarily go between the `<script>` tag can go in your external file
- Cannot have the `<script> </script>` tags themselves in the file
- Once loaded, the script will hang around in the browser's cache (no need to load the same script twice)

JS is Case Sensitive

- `var id` is not the same as `var ID` or `var iD`
- JS is also a camel-cased language
  - `getElementById`
  - First letter uncapitalized and capitalize the first letter of each word
- By contrast, HTML itself is NOT case sensitive

Output (`writeln`)

```
<html>
  <head>
  </head>
  <body>
    <script type='text/javascript'>
      document.writeln('Hello World!');
    </script>
  </body>
</html>
```

- Use this only while the page is loading
- If used after the page has loaded, the browser will destroy the page and start constructing a new one

Output (`alert`)

```
<html>
  <head>
  </head>
  <body>
    <script type='text/javascript'>
      alert('Hello World!');
    </script>
  </body>
</html>
```

- Useful for debugging
- Showing an annoying alert box
- Stop script running until the user clicks the OK button

Output: print to console

```
<html>
  <head>
  </head>
  <body>
    <script type='text/javascript'>
      console.log('Hello World!');
    </script>
  </body>
</html>
```

- Useful for debugging
- Print to console not the browser window

Output (`getElementByID`)

```
<html>
<head></head>
<body>
  <div id='feedback'></div> // define a division
  <script type='text/javascript'>
   document.getElementById('feedback').innerHTML='<P><font
color=red>Hello World!</font>';
  </script>
</body>
</html>
```

- Can change the contents of `feedback` anytime, even after the page has finished loading
- `innerHTML` is not a published standard but widely used
- Can use full-blown HTML

Input (`onClick`)

```
<html>
<head></head>
<body>
  <div id='feedback' onClick='goodbye()'>Users without 
Javascript see this.</div>
  <script type='text/javascript'>
   document.getElementById('feedback').innerHTML='Hello 
World!';
   function goodbye(){
    document.getElementById('feedback').innerHTML='Goodbye 
World!';
    }
  </script>
</body>
</html>
```

Input (user input)

```
<html>
<head></head>
<body>
  <input id='userInput' size=60>
  <button onClick='userSubmit()'>Submit</button><BR>
   <P><div id='result'></div>
   <script type='text/javascript'>
     function userSubmit() {
       var UI=document.getElementById('userInput').value;
       document.getElementById('result').innerHTML='You
typed: '+UI;
     }
   </script>
</body>
</html>
```

Input (user input), w/o button

```
<html>
<head></head>
<body>
  <input id='userInput' onKeyUp="userSubmit()"
size=60><BR>
   <P><div id='result'></div>
   <script type='text/javascript'>
     function userSubmit() {
       var UI=document.getElementById('userInput').value;
       document.getElementById('result').innerHTML='You
typed: '+UI;
     }
   </script>
</body>
</html>
```

JS is an Event Driven Language

- Your scripts react to events you set up
- Your code waits until an event starts something up
- A short-list of common events
  - `onClick, onDblClick`
  - `onFocus, onSelect`
  - `onKeyDown, onKeyPress, onKeyUp`
  - `onLoad, onUnload`
  - `onMouseDown, onMouseMove, onMouseOut, onMouseOver, onMouseUp`
  - `onSubmit, onChange, onResize`

Comments

- `//`: ignore everything to the end of the line
- `/*  */`: ignore everything in between
  - Do not put `</script>` tag in the comment

```
/* The browser will break the Javascript when it
sees this </script> tag. Everything from tag forward
is now being processed as HTML! This is a bad thing!
To avoid this you need to avoid using this tag
anywhere in your Javascript. */
```

- Comments are a liability in JS since they will be transmitted along with the code to each and every page load
- Not a big problem for this class

Variables

- JS is not a strongly typed language
- Variables can store anything, even functions

```
var thisIsAString = 'This is a string';
var alsoAString = '25';
var isANumber = 25;
var isEqual = (alsoAString == isANumber); // This is true,
they are both 25
var isEqual = (alsoAString === isANumber); // False one is a
number, the other a string
var concat = alsoAString + isANumber; // concat is now 2525
var addition = isANumber + isANumber; // addition is now 50
```

Variables

```
var alsoANumber = 3.05; // is equal to 3.05
var floatError = 0.06+0.01; // is equal to 0.07
var anExponent = 1.23e+3; // is equal to 1230
var hexadecimal = 0xff; // is equal to 255 (15 * 16 + 15)
var octal = 0377; // is equal to 255 (3 * 64 + 7 * 8 + 7)
var isTrue = true; // This is a boolean, it can be true or
false
var isFalse= false; // This is a boolean, it can be true or
false
var isArray = [0, 'one', 2, 3, '4', 5]; // This is an array
var four = isArray[4]; // assign a single array element to a
variable, in this case four = '4'
```

Variables

```
// This is a Javascript object
var isObject = { 'color': 'blue',
    'dog': 'bark',
    'array': [0,1,2,3,4,5],
    'myfunc': function () { alert('do something!'); }
}
var dog = isObject.dog; // dog now stores the string 'bark'
isObject.myfunc(); // creates an alert box with the value "do
something!"
var someFunction = function() { return "I am a function!"; }
var alsoAFunction = someFunction; // No () so alsoAFunction
becomes a function
var result = alsoAFunction(); // alsoAFunction is executed
here because () executes the function so result stores the
return value of the function which is "I am a function!"
```

Variables

- Functions themselves can be defined like, and act like variables
- Once defined, a function can be passed to other functions as an argument, or assigned to other variables just like a string, array or any other JS object
- Use function w/o `()`, the function is treated like a variable and can be passed and assigned
- Use function w `()` invoke the function, executing it and passing back the return value (if any)

Variable Scope

- All variables are global unless explicitly defined inside a function
- A function defines a new variable w/o using the `var` keyword, that variable will be global in scope!

`"use strict"`

- Define that JS code should be executed in “strict mode”
- New in JS 1.8.5
- This literal expression is added to the beginning of a JS file or a JS function
  - Declared at the beginning of a JS file, it has global scope (all code will execute in strict mode)
  - Declared inside a function, it has local scope (only the code inside the function is in strict mode)

Examples

```
"use strict";
x = 3.14;         // This will cause an error
```

```
"use strict";
myFunction();
function myFunction() {
    y = 3.14;   // This will also cause an error
}
```

```
x = 3.14;         // This will not cause an error
myFunction();
function myFunction() {
   "use strict";
    y = 3.14;   // This will cause an error
}
```

Why Strict Mode

- Make it easier to write “secure” JS
- Changes previously accepted “bad” syntax: into real errors
  - In normal JS, mistyping a variable name creates a new global variable
  - In strict mode, this will throw an error, making it impossible to accidentally create a global variable
- Please add `"use strict”;` to the beginning of your JS code!

Reserved Words

abstract	boolean	break	byte
case	catch	char	class
const	continue	debugger	default
delete	do	double	else
enum	export	extends	final
finally	float	for	function
goto	if	implements	import
in	instanceof	int	interface
long	native	new	package
private	protected	public	return
short	static	super	switch
synchronized	this	throw	throws
transient	try	typeof	var
void	volatile	while	with

Special Keywords

- `NaN` – not a number (generated when an arithmetic operation returns an invalid result)
  - `isNaN(3/'dog')`
- `Infinity` (returned when an arithmetic operation overflows JS's precision)
- `Null` means “empty” and is evaluated to false when used in boolean operation
- `true` and `false` as boolean value

Arithmetic Operations

- `+ , - , * , / , % , ++ , --`

```
var x = 5;
var y = x++; // y=5, x=6
```

```
var x = 5;
var y = ++x; // y=6, x=6
```

Logical and Comparison Operations

- = assignment
- == equality
- === identity, check value and data type
- != not equal
- !== not identical
- ! not
- || or
- && and
- < , <= , > , >=

Conditionals: `if/else`

```
var x=5;
if (x==1) {
   alert('x is equal to 1!');
} else if (x==2) {
   alert('x is equal to 2!');
} else if (x==5) {
   alert('x is equal to 5!');
} else {
   alert('x isn't 1, 2 or 5!');
}
```

Conditionals: `switch`

```
var x=5;
switch (x) {
   case 1: alert('x is equal to 1!'); break;
   case 2: alert('x is equal to 2!'); break;
   case 5: alert('x is equal to 5!'); break;
   default: alert('x isn't 1, 2 or 5!');
}
```

Conditionals: Shorthand Assignment

```
function doAddition(firstVar, secondVar) {
   var first = firstVar || 5;
   var second = secondVar || 10;
   return first+second;
}
doAddition(12); // return 22, as firstVar is assigned
but not secondVar
```

- Use a logical OR to determine if the passed variables actually have a value
- The first variable `firstVar` is a non-falsey value (actually defined) but the second variable `secondVar` is not (undefined)

Conditionals: Ternary Operators

```
var userName = 'Bob';
var hello = (userName == 'Bob') ? 'Hello Bob!' :
'Hello Not Bob!’;
alert(hello); // 'Hello Bob!'
```

Loops: `for`

```
for (var i=0; i<5; i++) {
   document.writeln('I is equal to '+i+'<br>');
}
// outputs:
// I is equal to 0
// I is equal to 1
// I is equal to 2
// I is equal to 3
// I is equal to 4
```

Loops: `for/in`

```
var myObject = { 'animal' : 'dog',
   'growls' : true,
   'hasFleas': true,
   'loyal' : true }
for (var property in myObject) {
document.writeln(property + ' contains ' +
   myObject[property]+'<br>');
}
// Outputs:
// animal contains dog
// growls contains true
// hasFleas contains true
// loyal contains true
```

Loops: `while`

```
var x = 1;
while (x < 5) { x = x + 1; }
```

```
var x = 1;
while (true) {
  x = x + 1;
  if (x >= 5) { break; }
}
```

```
var x = 1;
do { x = x + 1;
} while (x < 5);
```

JS Notes

- Is JS slow?
  - JS engines in browsers are getting much faster
  - Not a key issues for graphics since once we get the data to the GPU it doesn’t matter how we got the data there
- JS is a (too) big language
  - We don’t need to use it all
  - Choose parts we want to use
  - Don’t try to make your code look like C or Java

References

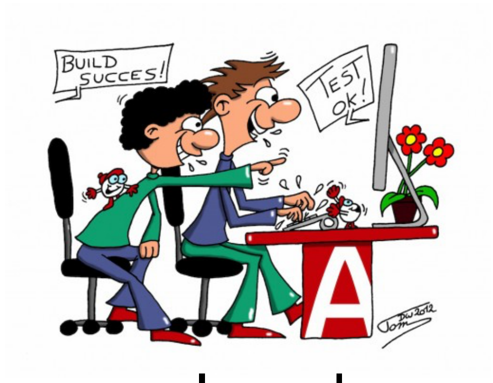
- *JavaScript: The Definitive Guide*, by David Flanagan, O'Reilly Media (ebook available via ND library)
- *JavaScript, The Good Parts*, by Douglas Crockford, O'Reilly Media
- Many web tutorials

A Minimalist Approach

- We will use only core JS and HTML
  - No extras or variants
  - No additional packages
    - CSS
    - JQuery
- Focus on graphics
  - Examples may lack beauty
- You are welcome to use other variants as needed

How to Debug WebGL/JS Code?

- All browsers have built-in features to support code debugging
- Advanced tools
  - Firebug and developer tools for HTML/JS (works with Firefox)
  - WebGL Inspector for WebGL (works with Chrome)
- For simplicity and consistency, we ask you to always use Firefox + Firebug for this class



Exercise

- Implement a bubble sort algorithm
  - Given an array of size n, take $(n\text{-}1)$ passes
  - For each pass, scan from the beginning of the array, compare two neighboring elements in a pair, and bubble down the larger of the two
  - So after the i-th pass, the $(n\text{-}i)$th element will be in its final position
- Use the template given

Homework

- Get Firefox and Firebug installed in your laptop by following the WebGL programming notes posted
- Go through the example code in this lecture and get yourself familiar with the basics of JavaScript coding