WEEK 6 | PROGRAMMING
Introduction to Javascript

Shawn Park & Jeff Zhan

{ } Web Design DeCal

DESIGN MEETS PROGRAMMING

Today's Outline

1. Intro to JavaScript
2. Basic Syntax
3. Conditional Statements
4. Loops

Goal Today:

Learning JavaScript

Intro to JavaScript

Intro to JavaScript

- JavaScript is the scripting language of the web
- **Not** the same as Java!
- Like CSS, you can select elements and *manipulate* their actions
  - Hide/Unhide menus
  - Photo Slider (**carousel**)
  - Popup modals (like Facebook's Photo Viewer)
  - Form Validation
  - One-Scroll Homepage

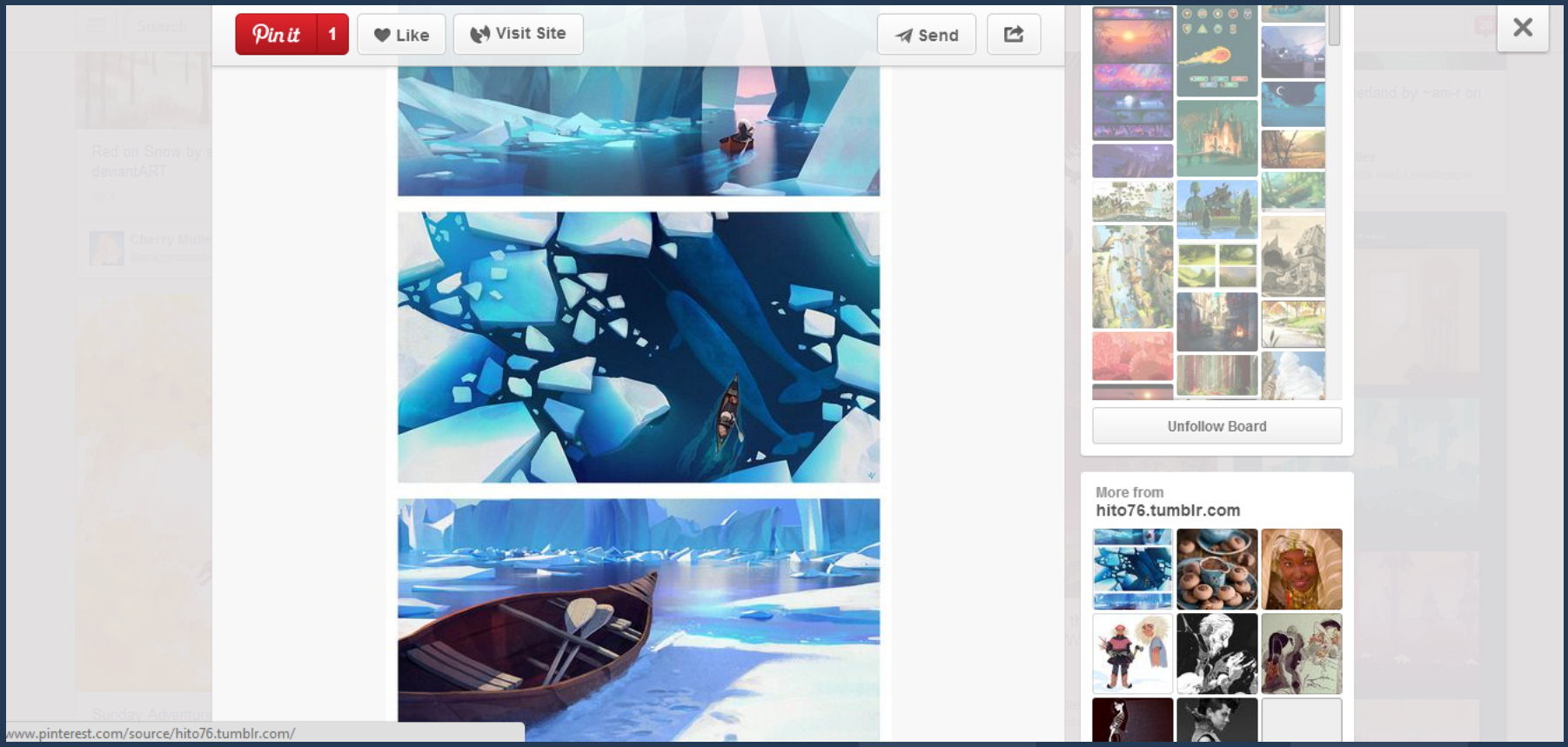
Intro to JavaScript

Sample JavaScript:

```
// Syntax highlighting
function printNumber()
{
  var number = 1234;
  var x;
  document.write("The number is " + number);
  for (var i = 0; i <= number; i++)
  {
    x++;
    x--;
    x += 1.0;
  }
  i += @; // illegal character
}
body.onLoad = printNumber;
```

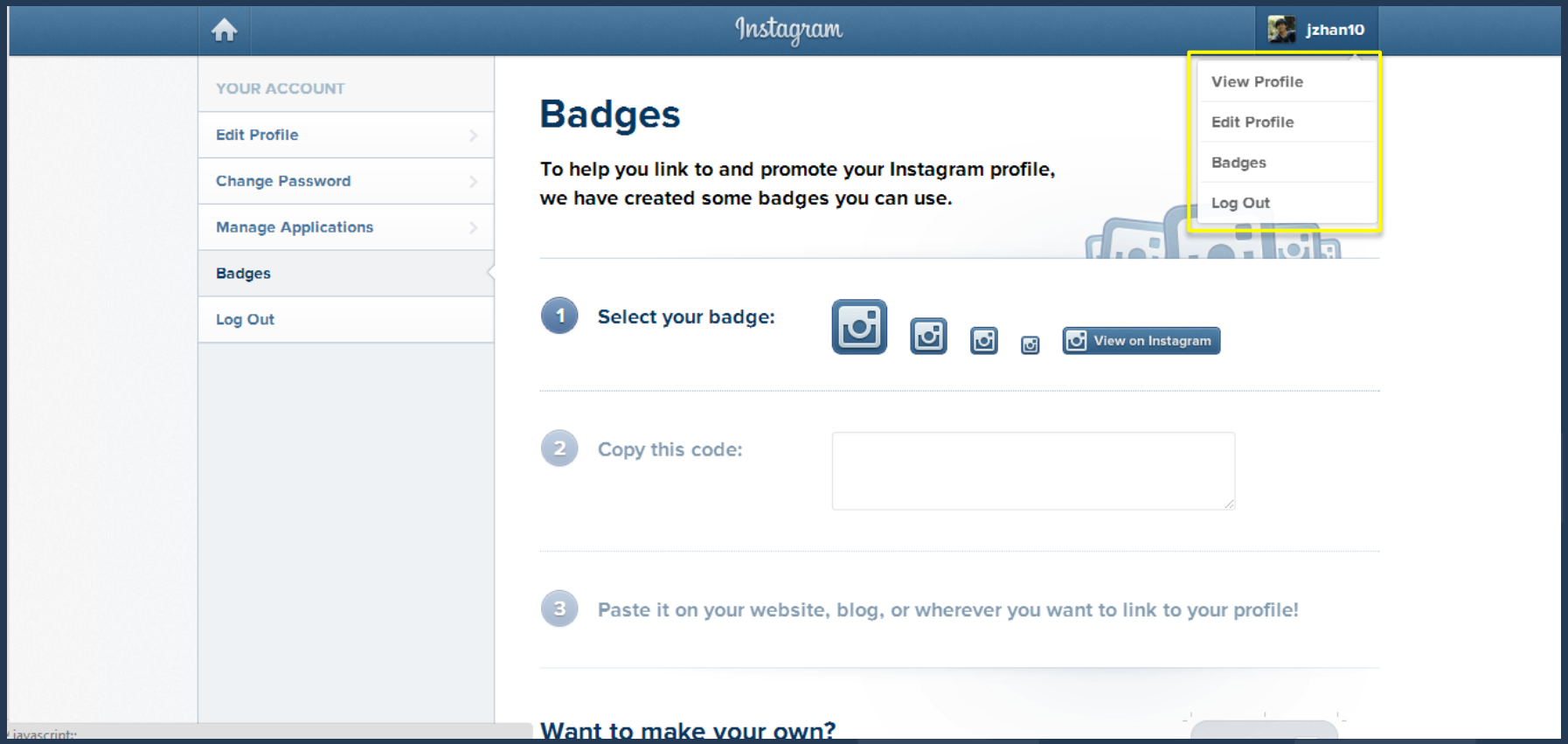
```
Accordian.prototype = {
    init: function( opts ) {
        this.defaults = {
            speed: 200,
            closeAll: true
        };

        this.options = $.extend( this.defaults, opts || {} );
        this.build();
    },

    build: function() {
        var self = this;

        $(function() {
            self.triggers = $( '.accordian-trigger' );
            //self.containers = $( '.accordian-container' );
            self.events();
        });
    },
```

Intro to JavaScript - Examples



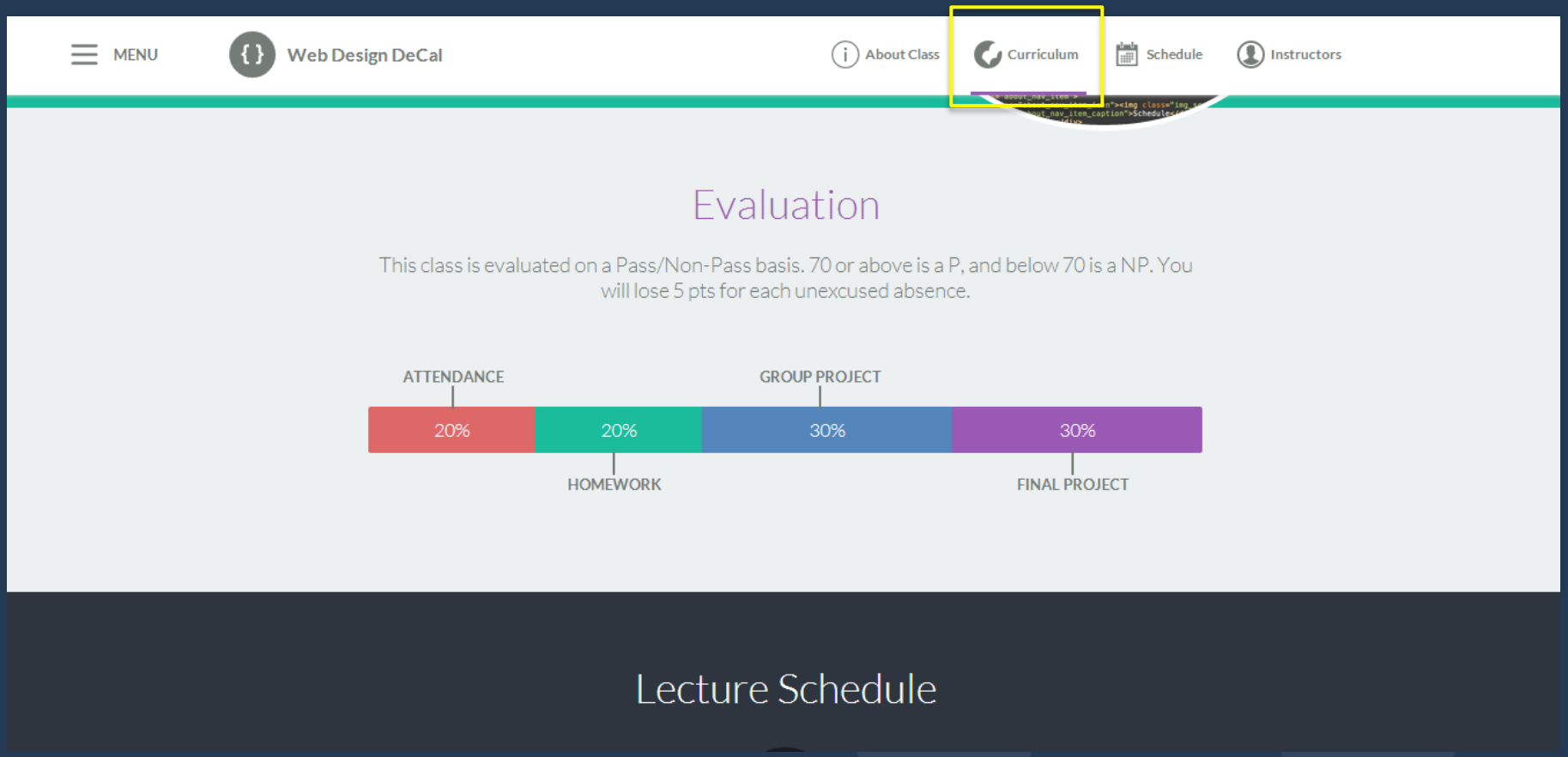
Pinterest - Modals

Intro to JavaScript - Examples



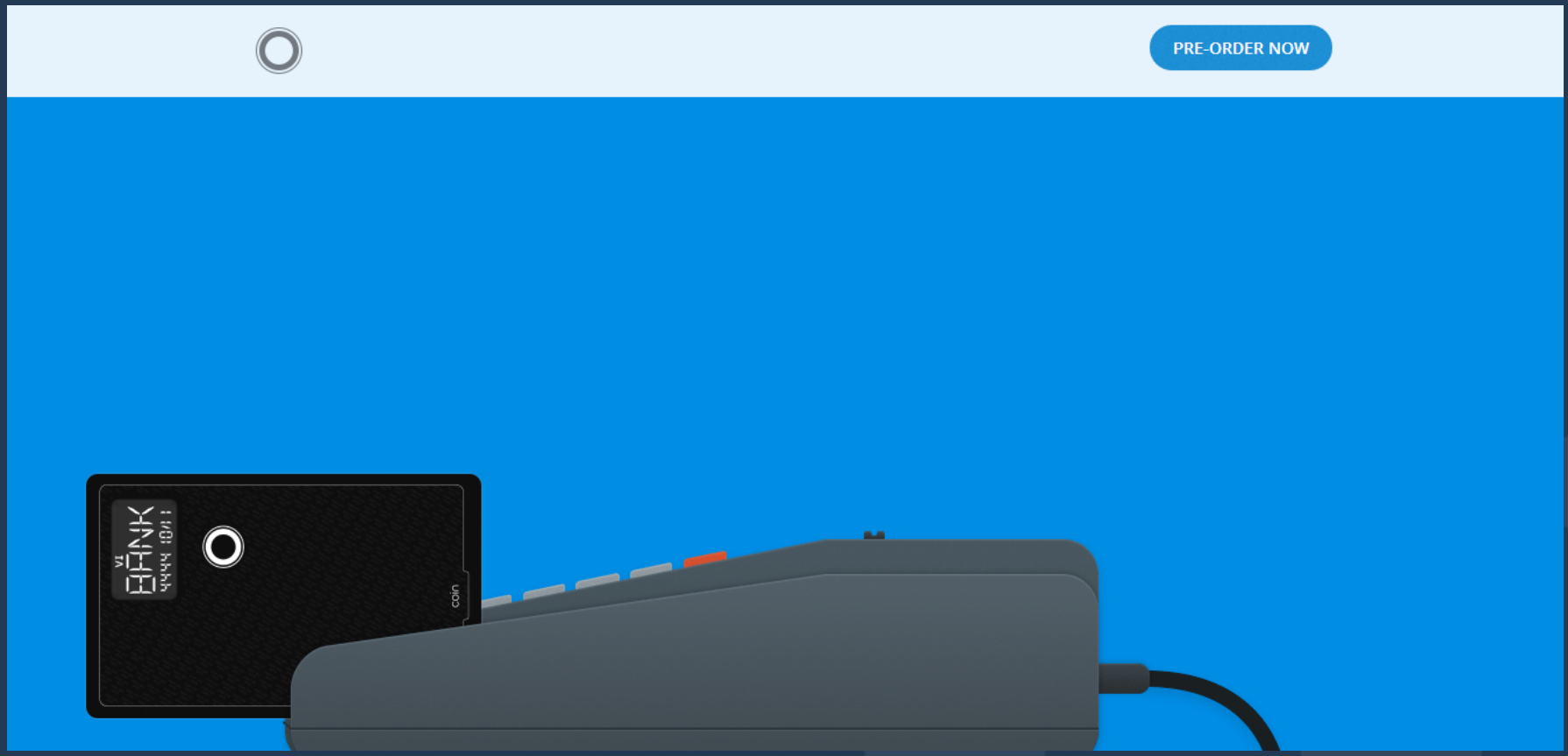
Instagram - Toggling Menus

Intro to JavaScript - Examples



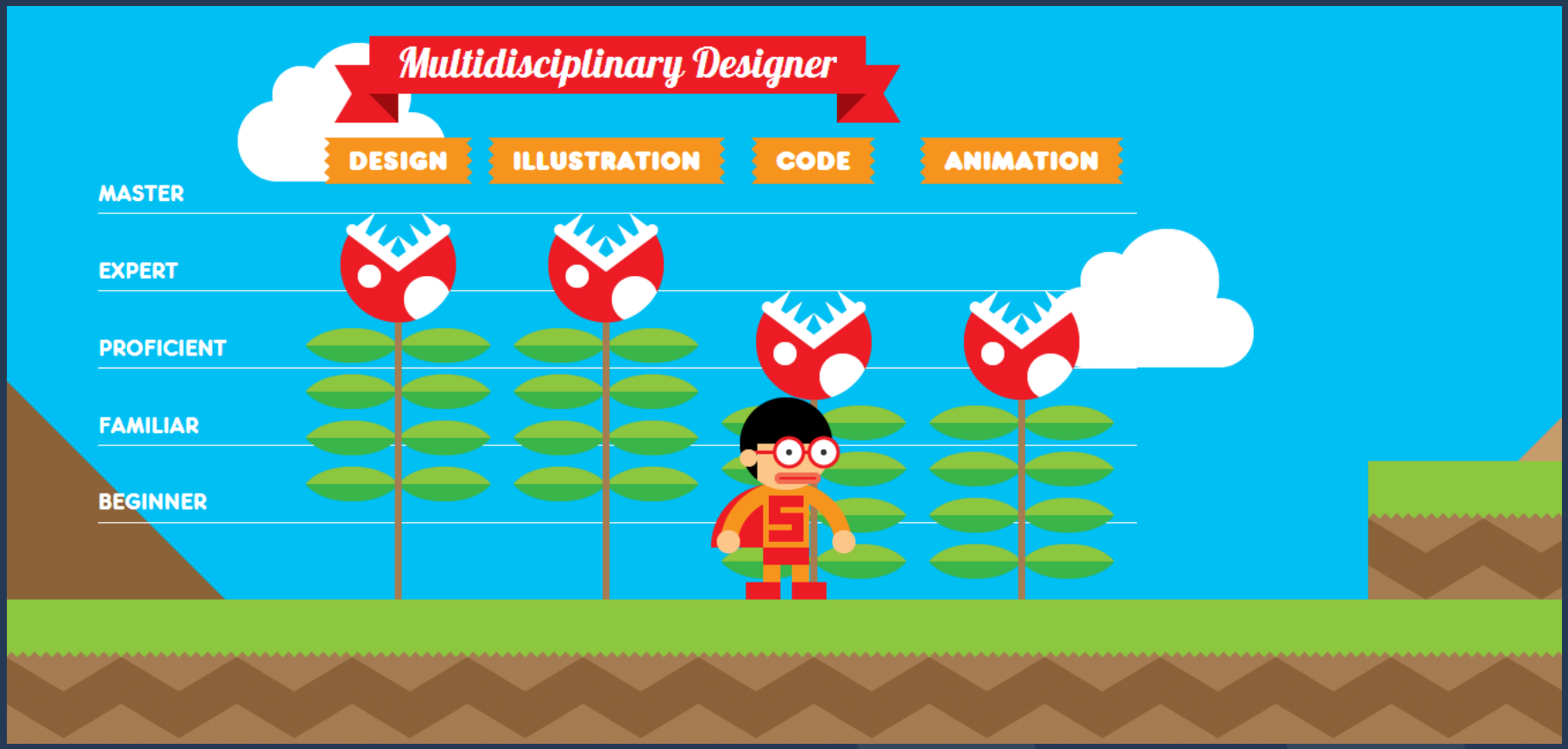
Decal Site – Menu Changes with Scroll

Intro to JavaScript - Examples



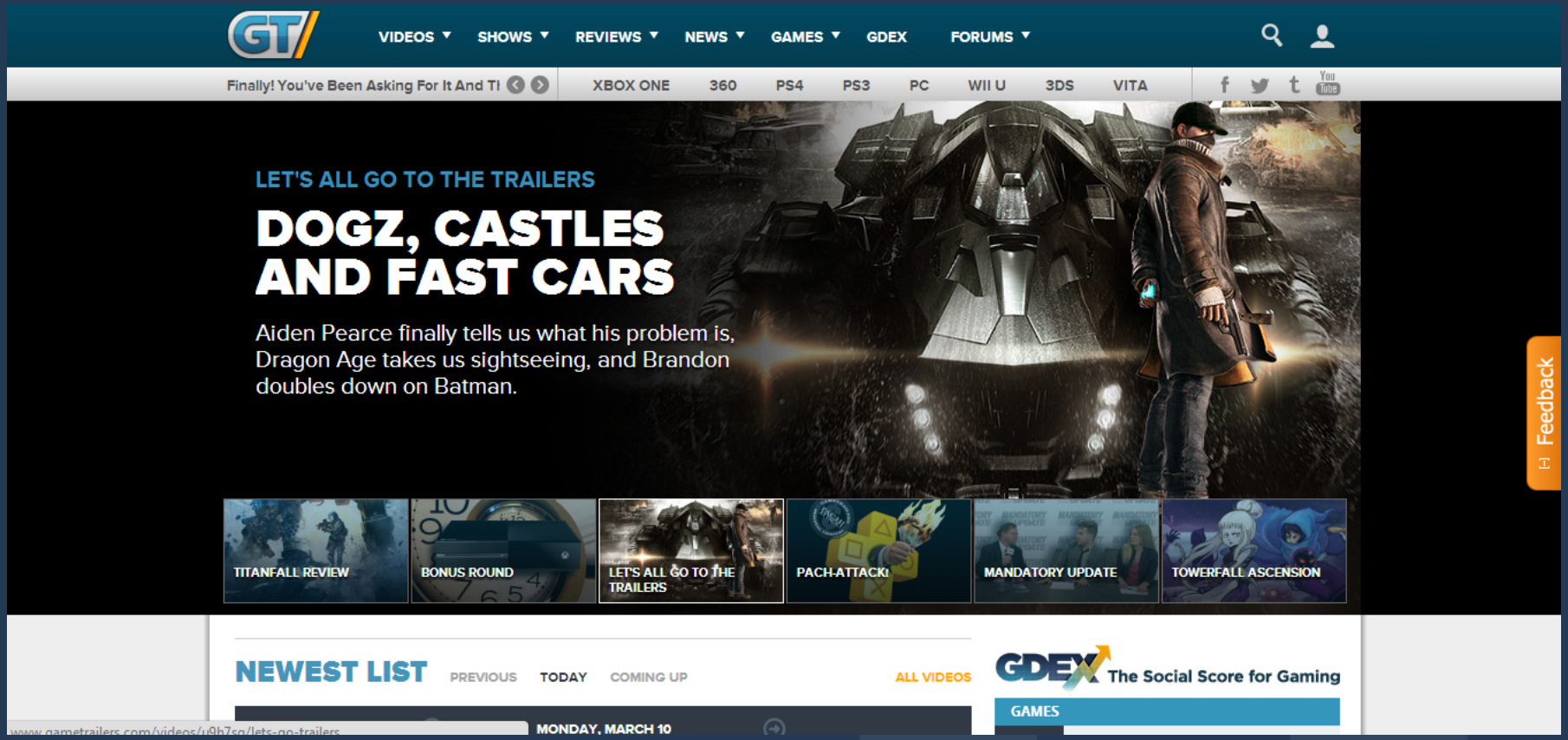
Coin – JavaScript animations while scrolling

Intro to JavaScript - Examples



Interactive Resume *(http://www.rleonardi.com/interactive-resume/)*

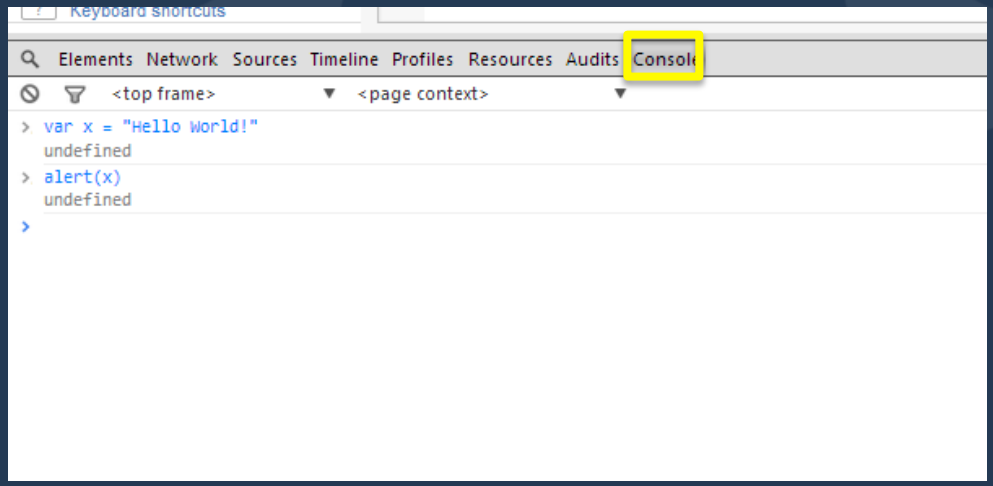
Intro to JavaScript - Examples



GameTrailers – Photo Slider

Intro to JavaScript

- How to you use Javascript?
- Like CSS, you link to a separate *.js* file that contains your file
  - *<script type="text/javascript" src="home.js"></script>*
- You can also **test** JavaScript in Chrome's **console** (Inspector Element window) and on JSFiddle.net
- Let's start coding!



Basic Syntax

- Variables
  - They take in numbers, text, booleans (true/false) and functions

```
var x = 8;
var y = "John"
var z = 'Smith'
var a = true;
```

Text (in computer science, we call them "strings") can be surrounded by single or double quotes

- Concatenation
  - You can add numbers or combine "strings"

```
var a = x + x;
var b = y + " " + z;
var c = 5 + " five";
```

"a" now stores 16. "b" stores John Smith.

Note: only use "var" to initialize (if the variable name has never been used before)

Basic Syntax

- Shorthand increment/decrement

```
var x = 1;

x = x + 1;
// x is now 2
x += 1;
// x is now 3
x++;
// x is now 4
x--;
// x is now 3
```

To increment by 1, there are 3 ways:

1) x = x+1
2) x += 1
3) x++

Same goes for decrementing

Conditionals

- If… Else…
  - If something do this, else do that

```
If(x > 9000) {
    alert(“Over 9000!”);
} else {
    console.log(“Weak”);
}
```

alert(…) is a JavaScript function that creates a popup.
console.log(…) outputs whatever is inside onto the console (e.g. Chrome’s console).
Both useful for debugging.

- Plain If, and If…Else If… Else If…

```
if(z < 9000) {
    alert(“Not Saiyan”);
}
```

```
if(z == 1000) {
        …
} else if (z == 2000) {
        …
```

Conditionals

- Comparisons
  - <, >, >=, <=
  - == and === (3 equal signs checks value **and** type)

```
var x = 3;
x == 3 // true
x === “3” // false
x === 3 // true
```

- And/Or - &&/||

```
if((x==3) && (y==5)) {
        …
```

```
if(x || y) {
        ….
```

Conditionals

- Not Equal - !=, !== (not same type too)

```
if(x != y) {
        …
}
```

- Remember, you can use ( ) parentheses to separate your logic. (*e.g. if((x != 3) && (y != 4)))*

Arrays

- Variables can also store more than 1 value

```
var x = new Array("Porsche", "BMW", "Ferrari");

                or

var x = ["Porsche", "BMW", "Ferrari"];

                or

var x = new Array();
x[0] = "Porsche";
x[1] = "BMW";
x[2] = "Ferrari";
```

Arrays

- Access an array with […]
  - First value is always the array position. Positions start at 0

```
var x = [“Porsche”, “BMW”, “Ferrari”];
alert(x[0]); // Will alert “Porsche”
```

- Set values for an array

```
x[2] = “Tesla”;
```

- You can get an array’s length with arrayname.length

```
x.length // returns 3. However, max position is 2, since we start at 0
```

Loops

- For Loop
  - Loop through items/arrays
  - 3 components
    - *for(var i=0; i < 10; i++)*
  - *1st value initializes variable. i++ increments i by 1 after loop is complete. Loop while i < 10, and stop when condition is not met.*

```
var cars = [“Porsche”, “BMW”, “Tesla”];
for(var i = 0; i < cars.length; i++) {
        alert(cars[i]);
}
// alerts each value in cars, one by one
```

Loops

- While Loop
  - Loop through items/arrays
  - 1 component
    - *while(i < 30) { … }*
  - Initialize (var i = 0) before while loop, increment (i++) inside

```
var i = 0;
while(i < 30){
        alert(i);
        i++;
}
// alerts values 0 to 29
```

Warning!

If you don't increment, or don't fail the conditional statement at some point, you end up in a never-ending loop! Not good!

Loops

- For vs. While
  - For loops are more compact
  - But while loops are fast to type

```
for(var i=0; i<100; i++) {
        alert(i);
}
```

```
var i = 0;
while(i < 100) {
        alert(i);
        i++
}
```

Output: 0, 1, 2, … , 98, 99

Loops

- For In Loops
  - For loops **designed for** arrays

```
var states = [“New York”, “Florida”, “Texas”, “California”];

for(state in states) {
        alert(state);
}
```

Always in format “... In …”. First value is the current element in the array, starting from position 0, and loops through the array 1 by 1. Extremely popular and useful!

Functions

- **Functions** are a set of JavaScript code, that can perform some action, return a result, or do computation
- They take in 0 or more **arguments** (values to be passed in)
  - Arguments are by default type **var**, so *arr* can be any **var**

```
function printOddValues(arr) {
        var odd = true;
        for(element in arr) {
                if(odd){
                        console.log(element+" ");
                        odd = false;
                } else {
                        odd = true;
                }
        }
}
```

Once you have a function, you can call it!

```
var arr = [1, 2, 3, 4, 5, 6, 7];
printOddValues(arr);
```

Output: 1 3 5 7

```
var arr = ["OH", "CA", "PA", "NY"];
printOddValues(arr);
```

Output: OH PA

Functions

- **Functions** can return values, and you can save them in a variable (for future use)

```
function calculateExp(value, times) {
        if(times < 0) {
                return -1;
        } else if(times == 0) {
                return 1;
        } else {
                var count = 1, total = 0;
                while(count <= times) {
                        total += value;
                }
                return value;
        }
}
```

Let's call this function!

```
var x = calculateExp(5, 3);
alert(x);
```

Output: 125

```
var x = calculateTimes(10, -999);
if(x != -1){
        alert("No neg nums please");
}
```

Output: "No neg nums please"

Demo

Summary

- Syntax
  - var takes in numbers, strings, booleans, and functions
- Conditionals
- Loops
  - for & while
- Functions
  - Set of JavaScript code
- Accessing HTML
  - getElementById
  - getElementsByClassName

All lecture material, handouts, and homework can be found at:
http://www.thewebdesignworkshop.co

Bonus Slides

(for those eager to start jQuery early)

jQuery

- Functions and variables are cool… but can we do more with JavaScript?
- **jQuery,** a JavaScript library, comes in handy!
- jQuery allows you to access HTML elements and do all sort of things!
  - Click events
  - Keyboard presses
  - Scrolling animations
  - Modal popups
  - Toggling menus
  - Smoother hovers and animations
  - Etc.

jQuery

- How do we add jQuery to our page?
- Either insert the first element below, or download the jQuery .js file and link to it
  - **Important!** Add this before any links to your own .js files!

```
<script src="http://code.jquery.com/jquery-1.11.0.min.js"></script>
```

Above: Link to an online copy of jQuery

```
<script src="assets/js/jquery-1.11.0.min.js"></script>
```

Above: Link to a local copy of jQuery

jQuery

- Initializing jQuery
  - Now that we linked to the jQuery library, we can use it in our own .js files
  - To start using jQuery, use the following syntax:

```
$(document).ready(function(){

        … your javascript here …

});
```

$ - Calls the jQuery library
(document) – Selects the entire HTML page
.ready – When *all* the HTML is loaded…
function(){ … } – …do everything inside this function

jQuery

- Sample

```
$(document).ready(function(){

    function add(x, y) {
        return x + y;
    }

    var total = add(3, 4);
    alert(total); // should open a popup with the number 7

});
```

jQuery

- The **heart of jQuery** is accessing HTML elements
- Like **$(document)**, we use this same syntax **$(…)**
  - Except in …, we replace with the element's CSS selector
  - Example:

```
$('body')
$('.box')
$('#container .box')
$('a')
```

Essentially you use put in CSS selectors, wrapped in quotes, into $(…) to select an element

- Once you select an element, you can access different values, properties, or functions to the element

jQuery – Accessing Content

- 3 common ways to access element content
  - **.text()** – Grabs all text inside an element
  - **.html()** – Grabs all HTML inside an element
  - **.value** – Grabs the value attribute or an element (useful for inputs)

```
<div id="box">
    <div class="title">
        Hello World!
    </div>
</div>
```

$('#box").text() → Hello World!

$('#box").html() →
Hello World!

- You could test the above using **console.log($('#box').text())** or **alert($('#box').html());**

jQuery – Accessing Content

- Value
  - Useful to get search results, form input, etc.

<input type="text" id="name"> $('#name').value → Jeff

- Note: Not only can you get values, you can also **set** them, using =
  - *E.g. Set new HTML: $('#box').html(' Yo ');*
  - *Set text in input box: $('#name').value = "Billy";*

jQuery – Event Listeners

- **Event Listeners** are always on the page, “listening” for an event to occur (and react to these events)
- Events include:
  - Clicking
  - Key presses
  - Scrolling
  - Hovering (smoother than CSS)
  - Tons more

jQuery – Event Listeners

- Event Listeners are *attached* to jQuery selected elements
  - *E.g. $('#box').click(function() { … });*
- Syntax:
  - .click(function(){ … });
  - .hover(function(){ … });
  - .ready(function(){ … }); // Remember this one?
- Always pass a function(){ … } to an event listener! (*event handlers*)

```
$('#box').click(function(){
        alert("You are so cool.");
});
```

Browser binds **click** event to **#box**. It continues to "listen" as user is on page, and when user clicks on #box, a popup will appear!

jQuery – Event Listeners

- More examples

```
$('#box').hover(function(){
        alert($(this).html());
});
```

Inside your event listener function, you can refer to the current element (say **#box**) with simply **$(this)**.

```
$('.box').hover(function(){
        alert($(this).html());
});
```

This hover now applies to all class **box**. **$(this)** only refers to the element you hovered on, not to all elements with class box!

```
$('.box').click(function(){
        $(this).html('Replacing');
});
```

Click on any element with class **.box**, and you replace all HTML code inside the element with the text 'Replacing';

jQuery – Event Listeners

- Other useful events:
  - **.toggle(function(){ … }, function(){ … });**
    - Essentially click, but clicking once does one thing, clicking again does another
    - Useful for toggling menus

```
$('#menu-icon').toggle(function(){
        $('#menu').show();
}, function(){
        $('#menu').hide();
});
```