# Problem #1: Responsive Design

Create a 3-step registration page.

## Guidelines

1. Step 1
   1. Contains the following fields: email, first\_name, username, password.
   2. Username validation:
      1. Minimum 4 letters.
      2. Maximum 12 letters.
   3. Password validation:
      1. Minimum 6 characters.
      2. Maximum 12 characters.
      3. Alphanumeric.
      4. One capital letter.
      5. One lower-case letter.
      6. One number OR special character: !@#$%^&\*()\_+-=
   4. Contains a checkbox with the text: “By checking this box, I agree to Terms & Conditions of the site.”
2. Step 2

Upon completion, the next step should show the following text and buttons.

* 1. Text: “You’re ready to play!  
     In order to play with real money, you will need to provide more information. You can do so now if you wish, or skip and do it at a later time.”
  2. Button 1: **Continue Registration** – Goes to Step 3.
  3. Button 2: **Skip** – Goes to: <http://www.casino.com/>.

1. Step 3
   1. Contains the following fields: Address 1, Address 2, City, Country, Postal Code, Phone Number, Bonus Code, Checkbox.
   2. Bonus Code should be hidden by default and become visible when clicking on a link with the following text: “Do you have a bonus code? Click here.”
2. Step 4

Upon completion, the next step should include the following text and buttons.

* 1. Text: “Thank you  
     You can now play with real money. Have fun, and good luck!”
  2. Button 1: **Play Now** – Goes to: <http://www.casino.com/casino-games/>
  3. Button 2: **Promotions** – Goes to: <http://www.casino.com/promotions/>

## Notes

1. The source code must be managed on an online-accessible VCS solution. (Such as [GitHub](https://github.com/).)
2. The design must be accessible from a public location. ([jsdo.it](http://jsdo.it/) or [Cloud9](https://c9.io/) are good options.)
3. The design should be responsive. It must adapt to all devices and screen sizes and work in both vertical and horizontal alignments.
4. Design it however you like, but make sure the design works well in all resolutions. (Bonus points for adding flare to specific device types.)
5. CTA buttons should be differentiated in design.
6. Add proper validations or limitations where needed.
7. Incorporate any animations, design characteristics you wish to the form elements, or page transitions.

# Problem #2: Broadcast System

Using Node.js (or an equivalent server architecture) please build a broadcast system.

## Guidelines

1. The system must have a front-end component that is comprised of a web page with the following:
   1. Username field
   2. Recipient field
   3. Message field
   4. Send button
   5. Output Area
2. On initial page load, the user will be prompted to provide a username (#1.a. above)
3. When entering a recipient and message, and submitting to the system, the following will happen:
   1. The server will log the message in a file and store it in the memory.
   2. The server will dispatch the message to the username provided in the recipient field.
   3. The recipient window will be updated with the new message. (1.e. above)
4. The system needs to handle the log file and prevent it from exceeding 100 bytes.

## Bonus points

1. Add a “Broadcast” button, which will send the message to all active users.
2. Add a “Send Receipt” functionality to the system, which will confirm to the user when a message was displayed on the recipient’s window.

# Problem #3: OOP

Please write a class structure (in the language of your choosing, such as: C#, JAVA, JavaScript, PHP) that describes the following objects and their properties/functions.

* Dolphin
  1. Age (property)
  2. Sleep (function that prints ‘Zzzzzzz’)
  3. Swim (function that prints ‘Splah’)
  4. Eat (function that prints ‘Num num num’)
* Lion
  1. Age (property)
  2. Sleep (function that prints ‘Zzzzzzz’)
  3. Roar (function that prints ‘Rahhhh!’)
  4. Attack (function that prints ‘POW!’)
  5. Eat (function that prints ‘Num num num)
* Eagle
  1. Age (property)
  2. Sleep (function that prints ‘Zzzzzzz’)
  3. Fly (function that prints ‘whoo Hooo!’)
  4. Attack (function that prints ‘POW!’)
  5. Eat (function that prints ‘Num num num)
* Bee
  1. Age (property)
  2. Sleep (function that prints ‘Zzzzzzz’)
  3. Fly (function that prints ‘Whoo Hooo!’)
  4. Eat (function that prints ‘Num num num)

# Problem #4: Cross-Domain Behavior

Assuming two pages exist under two different sub domains; one page is included inside the other one, using an iframe.

The parent page’s URL is: **a.mydomain.com**  
The inner page’s URL is: **b.mydomain.com**

a.mydomain.com

b.mydomain.com

## Question

Is it possible for the parent page to call a JavaScript function of the inner page?

Yes or no, please explain the answer.

# Problem #5: System Architecture

Create a design document for a system that will serve as a “Master” API for an unknown amount of 3rd-party services. This API will serve as a single point of contact from an internal system to multiple 3rd-party services.

## Guidelines

1. Create a document that includes the following sections:
   1. Architecture Diagram: A diagram showing all components of the system, including the relevant hardware/software specifications. (For example: “DB: MySQL”, or “DB: MongoDB”.)
   2. User Stories: Sample flows describing how the system will behave.
   3. Advantages and Disadvantages: A written overview of the advantages and disadvantages of the system and the architecture.

## Notes

1. The system must use a single protocol for its public end-points. Choose whatever protocol you see fit.
2. The system must, at the very least, be able to connect to REST and SOAP services.