HTML, CSS and PHP:

mobile-first.CSS:

```css
.product-info { font-size: 2em; }
.products { text-align: center; display: flex; flex-wrap: wrap; padding: 15px; }
.sale-header { text-align: center; }
#header { background-color: yellow; padding: 50px; font-size: 20px; position: sticky; top: 05px; }
section > h1 { font-size: 100px; }
.product-card { flex-grow: 1; }
.product-image { background-color: lightblue; padding: 50px; margin: 10px; border-radius: 10px;
}
.product-image > img { border-radius: 100%; max-width: 250px; min-width: 250px; max-height:
250px; min-height: 250px; }
@media screen and (max-width: 768px) { .product-info { font-size: 1.2em; } #header { padding:
15px; font-size: 16px; } .product-image { padding: 15px; } .product-image img { max-width:
200px; max-height: 200px; } }
@media screen and (max-width: 480px) { .product-info { font-size: 1em; } #header { padding:
10px; font-size: 14px; } .product-image { padding: 10px; } .product-image img { max-width:
150px; max-height: 150px; } }
```

index.php (A1)

```php
<?php
ini_set('display_errors', 'On');
require_once "model/FrogGame.php";
session_save_path("sess");
session_start();
$dbconn = db_connect();
$errors = array();
$view = "";

if (!isset($_SESSION['state'])) {
    $_SESSION['state'] = 'login';
}

if (isset($_GET['operation']) && $_SESSION['state'] != 'login') {
    $view = "template.php";
    $_SESSION['state'] = 'game';

    switch ($_GET['operation']) {
        case "logout":
            session_destroy();
            session_start();
            $view = "login.php";
            $_SESSION['state'] = 'login';
            break;
        case "frog":
            $_SESSION["game"] = "frogIndex.php";
            break;
    }
}

if ($_GET["operation"] == "register") {
    $view = "register.php";
    $_SESSION['state'] = 'register';
}

switch ($_SESSION['state']) {
    case "login":
        $view = "login.php";

        if (empty($_REQUEST['submit']) || $_REQUEST['submit'] != "login") {
            break;
        }

        $query = "SELECT * FROM appuser WHERE userid=$1 AND password=$2;";
        $result = pg_prepare($dbconn, "", $query);
        $result = pg_execute($dbconn, "", array($_REQUEST['user'], hash('sha1',
$_REQUEST['password'])));
        if ($row = pg_fetch_array($result, NULL, PGSQL_ASSOC)) {
            $_SESSION['user'] = $_REQUEST['user'];...
        } else {
            $errors[] = "Invalid login";
        }
        break;

    case "register":
        $view = "register.php";

        if (empty($_REQUEST['submit'])) {
            break;
        }

        if ($_REQUEST['submit'] == "Back To Login") {
            $view = "login.php";
            $_SESSION["state"] = "login";
            break;
        } elseif ($_REQUEST['submit'] == "Register") {
            if (empty($_REQUEST['username']) || empty($_REQUEST['password']) ||
empty($_REQUEST['verify_password'])) {
                $errors[] = "Username and both passwords are required";
                break;
            }
                                    more logic
          break;
                  }
}

require_once "view/$view";
?>
```

some crazy js wss

```javascript
function DOMComb (oParent, indent) {
console.log(indent+oParent.nodeName+" START");
if (oParent.hasChildNodes ()) {var childIndent=indent+"    ";
for (var oNode = oParent.firstChild; oNode; oNode = oNode.nextSibling) {
DOMComb (oNode, childIndent);}}
console.log(indent+oParent.nodeName+" END");}
onload = function () { DOMComb (document.body, ""); };
list.splice(index, 1);
wss.on('connection', function(ws) {
        ws.send(JSON.stringify({'list': list}));
        ws.on('message', function(data, isBinary) {
                const message = isBinary ? data : data.toString();
                        const req = JSON.parse(message)
                const command = req.command;
                if (command == "add") {
                        if (req.item != "") {
                                list.push(req.item);
                                wss.broadcast(JSON.stringify({'list': list}));
                        } else {
                                wss.broadcast(JSON.stringify({'message': 'Item must be non-emtpy'}));
                        }
                } else if (command == "delete") {
                        list.splice(req.index, 1);
                        wss.broadcast(JSON.stringify({'list': list}));
                }
        });
});
```

Just putting in view of play not won:

```php
<?php
                // So I don't have to deal with uninitialized $_REQUEST['guess']
                $_REQUEST['guess']=!empty($_REQUEST['guess']) ? $_REQUEST['guess'] : '';
?>
```

```html
<!DOCTYPE html>
<html lang="en">
                <head>
                                <meta charset="utf-8">
                                <title>Frog</title>
                </head>
                <body>
        <form method="post">
            <div class="frogs">
                <?php
                    foreach ($_SESSION["FrogGame"]->frogs as $key=>$value) {
                        if ($value==1) {
                            echo('<button name="submit" value='. $key . '> <img
src="images/yellowFrog.gif" alt="Yellow Frog" height=50px> </button>');
                        } else if ($value==2) {
                            echo('<button name="submit" value='. $key . '> <img
src="images/greenFrog.gif" alt="Green Frog" height=50px> </button>');
                        } else {
                            echo('<button> <img src="images/empty.gif" alt="Empty" height=50px>
</button>') }}?>
            </div>

            <input type="submit" name="submit" value="reset" />
        </form>
                </body>
</html>
```

Frog Model

```php
<?php
class FrogGame {
    public $frogs = array(1, 1, 1, 0, 2, 2, 2);
    public $win = false;
    public function move($frog) {
        $type = $this->frogs[$frog];
        if ($type == 1) {
            if ($frog >= 6) return;
            if ($this->frogs[$frog+1] == 0){
                $this->frogs[$frog]=0;
                $this->frogs[$frog+1]=1;
            } else if ($frog < 5 && $this->frogs[$frog+2] == 0){
                $this->frogs[$frog]=0;
                $this->frogs[$frog+2]=1;
            }
        } else {
            if ($frog <= 0) return;
            if ($this->frogs[$frog - 1] == 0){
                $this->frogs[$frog]=0;
                $this->frogs[$frog-1]=2;
            } else if ($frog > 1 && $this->frogs[$frog-2] == 0){
                $this->frogs[$frog]=0;
                $this->frogs[$frog-2]=2;
            }
        }
        if ($this->frogs == array(2, 2, 2, 0, 1, 1, 1))$this->win=true;
    }
    public function reset() {
        $this->frogs = array(1, 1, 1, 0, 2, 2, 2);
        $this->win = false;
```

How session looks

```
state|s:4:"game";user|s:4:"user";game|s:15:"unavailable.php";stats|s:9:"stats.php";frogState|s:3:
```

CSS YAP:

Relative: This positions an element relative to its normal position in the document flow. It allows you to use the top, right, bottom, and left properties to adjust its position relative to where it would normally appear.
Absolute: This positions an element relative to its nearest positioned ancestor (an ancestor with a position value other than static), or relative to the initial containing block if no positioned ancestor is found. It's taken out of the normal document flow, meaning other elements will ignore it when determining their positions.

Fixed: This positions an element relative to the viewport, which means it stays fixed in its position even when the page is scrolled. It's also taken out of the normal document flow.
Sticky: This is a hybrid of relative and fixed positioning. The element is treated as relative positioned until it crosses a specified threshold (usually based on scrolling), at which point it becomes fixed. It then remains fixed until its containing block crosses a different threshold.

```
s1 s2 /* apply to s2 descendents of s1 */
s1>s2 /* apply to s2 children of s1 */
s1, s2, s3 /* apply to s1 and s2 and s3 */
```

Jquary, JS:

```
$sessionId = $_COOKIE['name'];
session_id($sessionId);
session_start();
$("#ui_home, #ui_username").hide();
$(ui).show();
$("#ui_home_button").removeClass("nav_selected");
$(`${ui}_button`).addClass("nav_selected");
.find .parents .children .append .prepend
$('.keyboardrow td').each(function() {
        var character = $(this).text().trim();
        alphabetMap[character] = this;
    });

var errorPopup = $('<div id="error-popup">' +
                        '<img src="icons/error.png">' +
                        '<h2>Oops!</h2>' +
                        '<p>' + message + '</p>' +
                      '</div>');

$('body').append(errorPopup);

errorPopup.fadeIn(400).delay(2000).fadeOut(400, function() {$(this).remove();});

$(key).css("background-color", "gray");

]var selector = `.row${gui_state.row} .col${gui_state.col}`;
$(selector).html('');

$('#games_won').text(wordle.won);
$('.keyboardrow td').click(buttonClick);

wordle = new Wordle(words);
class Wordle {
                    constructor(words){
                                        this.words=words;
}
document.getElementById("status").innerText = "You won!";
document.getElementById("status").appendChild(playAgainButton);
document.getElementById("status").innerHTML = "";
let buttons = document.querySelectorAll("button");
buttons.forEach(button => button.disabled = true);
<td><button onclick="pourCup(0)">Pour to other cup</button> </td>
```

Client setver HTTP:

```
telnet cslinux.utm.utoronto.ca 80
GET /~arnold/missing.html HTTP/1.1
Host: localhost
Set-Cookie: name=value [;EXPIRES=dateValue] [;DOMAIN=domainName] [;PATH=pathName]
[;SECURE]


HTTP/1.1 404 Not Found
Date: Thu, 23 Jan 2014 04:34:55 GMT
Server: Apache/2.4.6 (Ubuntu)
Content-Length: 292
Content-Type: text/html; charset=iso-8859-1

<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>404 Not Found</title>
</head><body>
<h2>Not Found</h2>
<p>The requested URL /~arnold/missing.html was not found on this server.</p>
<hr>
<address>Apache/2.4.6 (Ubuntu) Server at localhost Port 80</address>
</body></html>
```

Summary of commands:

GET a specified resource
DELETE a specified resource
HEAD get the responce headers (without associated data)
PUT accompanying data at the specified location
POST accompanying data as a subordinate to the specified location.

Some useful JS:

getElementById()	Returns the element that has the ID attribute with the specified value}
getElementsByTagName()	Returns a NodeList containing all elements with the specified tagname
getElementsByTagNameNS()	Returns a NodeList containing all elements with the specified namespaceURI and tagname
getElementsByName()	
getElementsByClassName()	
querySelector()	
querySelectorAll()	

Backend of wordle using node and express:

```
var port = 8662;
var express = require('express');
var crypto = require('crypto');
var app = express();
const fs = require('fs');
const Wordle = require('./model.js');

const wordleModels = {};
var words = [];


app.put('/wordle/:username/reset', function(req, res) {
    var username = req.params.username;
    if (wordleModels[username]) {
        wordleModels[username].reset();
        res.status(200).json({});
    } else {
        res.status(404).json({ error: 'User not found' });
    }
});


// https://scotch.io/tutorials/use-expressjs-to-get-url-and-post-parameters
app.use(express.json()); // support json encoded bodies
app.use(express.urlencoded({ extended: true })); // support encoded bodies

// https://expressjs.com/en/starter/static-files.html
app.use(express.static('static-content'));

app.listen(port, function () {
                    console.log('Example app listening on port '+port);
});
```

Using Ajax in the controller:

```
// As a result of the latest guess, update the colours of the game board
// and keyboard

function colourBoardAndKeyboard(score) {
    // Apply colors based on the score
    for (var i = 0; i < score.length; i++) {
        var item = score[i];
        var scoreColor = '';
        if (item.score === 3) {
            scoreColor = 'green';
        } else if (item.score === 2) {
            scoreColor = 'yellow';
        } else if (item.score === 1) {
            scoreColor = 'grey';
        }
        var rowIndex = gui_state.row;
        var letterIndex = i % 5;
        $('.row' + rowIndex + ' .col' + letterIndex).css('backgroundColor', scoreColor);
        $('.keyboardrow td:contains("' + item.char + '")').css('backgroundColor', scoreColor);
    }
    $('.keyboardrow td:contains("DEL")').css('backgroundColor', 'black');
    $('.keyboardrow td:contains("ENTER")').css('backgroundColor', 'black');



function updateWins() {
    $.ajax({
        method: "GET",
        url: "/wordle/" + username + "/wins",
        dataType: "json"
    })
    .done(function(data) {
        var winsElement = document.getElementById('wins');
        winsElement.textContent = data.wins;
    })
    .fail(function(err) {
        console.log("fail " + err.status + " " + JSON.stringify(err.responseJSON));
    });
}


/******************************************************************************
 * onload
 ******************************************************************************/
$(function() {
    $.ajax({
        method: "POST",
        url: "/wordle",
        dataType: "json"
    }).done(function(data) {
                                        username = data.username;
        $("#username").html(username);

        gui_gameEnable();
        showUI("#ui_username");
    }).fail(function(err){
                                        console.log("fail "+err.status+"
"+JSON.stringify(err.responseJSON));
                    });
});
```

More php

```
echo "<table border='1'>";
echo "<tr><th>ID</th><th>Name</th><th>Email</th></tr>";
while ($row = pg_fetch_assoc($result)) {
echo "<tr>"; echo "<td>" . $row['id'] . "</td>"; …
echo "</tr>";  } echo "</table>";
```

Promices and Callbacks in Rest API

Callbacks are functions that are passed as arguments to other functions. Promises, on the other hand, are objects representing the eventual completion or failure of an asynchronous operation. They provide a cleaner and more structured way to handle asynchronous code,

using methods like .then() and .catch() to chain asynchronous operations sequentially or handle errors in a more readable manner.
They both provide a way to execute code once an asynchronous operation is completed.

```
function fetchData() {                                                   async function
fetchDataAsync() {
  return new Promise((resolve, reject) => {            try {console.log("Fetching data...");
    setTimeout(() => {                                                   const data = await
fetchData(); // Wait for the Promise to resolve
      const data = { message: "Data fetched successfully" }; console.log("Data:", data);
      resolve(data); // Resolve the Promise with the fetched data    return data; // Return the
fetched data
    }, 2000); // Simulating 2 seconds delay     } catch (error) {  console.error("Error fetching
data:", error);
  });                                                   throw error; // Throw the error for
handling further up the call stack}}

}
```

Web Sockets

Make Socket

```
const WebsocketServer = require('ws').Server;
const wss = new WebsocketServer({ port: port + 1 });

wss.on('connection', conn => {
  conn.on('message', (data, isBinary) => {
    const message = isBinary ? data.toString() : data; // mixed up in sampleServerSimple.js?
    const obj = JSON.parse(message);
    switch (obj.operation) {
      case 'add':
        myList.push(obj.data);
        return conn.send(JSON.stringify({ message: '' }));
      case 'delete':
        myList.splice(myList.indexOf(obj.data), 1);
        return conn.send(JSON.stringify({ message: '' }));
      case 'get':
        return conn.send(JSON.stringify({ list: myList })); } });
```

Connect to Socket

```
const conn = new WebSocket('ws://myserver.com:10001');
function getList() {
  conn.send(JSON.stringify({ operation: 'get' });}

function addMore() {
  conn.send(JSON.stringify({ operation: 'add', $('#item').val() });}
conn.addEventListener("message", ev => {
  const obj = JSON.parse(ev.data);
  if (obj.list) {
    const table = $("<table><tr><th>Values</th></tr></table>");
    for (const item of obj.list) {
      table.append($("<tr><td>" + item + "</td></tr>")); }
    $("#theList").html(table);}});
```

AJAX

```
await ajaxStats()
            .then(function(response) {
                stats = response;})
            .catch(function(error) {
                console.error("Error:", error);});

async function ajaxStats() {
    return new Promise(function(resolve, reject) {
        $.ajax({
            method: "GET",
            url: "/stats/" + username,
            success: function(response) {
                resolve(response);},
            error: function(xhr, status, error) {
                reject(error)} });});}
```

Web Sockets: NODE

```
var staticPort = 10000;
var webSocketPort = staticPort+1;
var express = require('express');
var app = express();

// static_files has all of statically returned content
// https://expressjs.com/en/starter/static-files.html
app.use('/',express.static('static_files')); // this directory has files to be returned

app.listen(staticPort, function () {
  console.log('Static content on port:'+staticPort);});

var WebSocketServer = require('ws').Server,wss = new WebSocketServer({port:
webSocketPort});
var messages=[];

wss.broadcast = function(message){
                    for(let ws of this.clients){
                                        ws.send(message); }
                    // Alternatively
                    // this.clients.forEach(function (ws){ ws.send(message); });}

wss.on('connection', function(ws) {
                    var i;
                    for(i=0;i<messages.length;i++){
                                        ws.send(messages[i]);}
        ws.on('message', function(data, isBinary) {
                const message = isBinary ? data : data.toString();

                                        console.log(message);
                                        // ws.send(message);
                                        wss.broadcast(message);
                                        messages.push(message);});});

wss.on('close', function(code, data) {
        const reason = data.toString();
        console.log('disconnected');});
```

Connect to Web Sockets: js

```
<!DOCTYPE html>
<html lang="en">
        <head>
                    <meta charset="utf-8">
                <script src="jquery-3.7.1.min.js"></script>
                <script>

var socket;
function send(){
    socket.send($('#message').val());
    $('#message').val("");}

$(function(){
    socket = new WebSocket(`ws://${window.location.hostname}:10001`);

socket.onopen = function (event) {
$('#sendButton').removeAttr('disabled');
                                        console.log("connected");};

socket.onclose = function (event) {
                    alert("closed code:" + event.code + " reason:" +event.reason + "
                    wasClean:"+event.wasClean);};

socket.onmessage = function (event) {
                    $('#messages').append("<br/>"+event.data);}});
</script>
</head>
<body>
<h3>Chat Console</h3>
<form>
                    <input type="text" id="message" />
                    <input type="button" id="sendButton" value="send" disabled='disabled'
onclick="send();" />
</form>
<div id="messages" style="border:1px solid black; width:100%; height:100px; overflow: auto;"
></div>
</body>
</html>
```

AJAX

```
$.ajax({
        method: "POST",
        url: "/new",
        success: function(response) {
            username = response.username;
            $("#username").html(username); },
        error: function(status, error) {
            console.error("Error:", status, error);}});
```

Connect to Web Sockets: React

```
componentDidMount() {
    this.fetchUsername();
    const socket = new WebSocket('ws://localhost:8001');
    socket.onopen = () => {
        console.log("Connected!");};
    socket.onclose = () => {
      console.log('Closed!'); };

    socket.onmessage = (event) => {
      console.log('Message received from server:', event.data);
      if (event.data === 'Restart') {
        this.handleNewGame();}
      if (!isNaN(event.data)) {
        this.setState({timer: event.data}); }};

    return () => {
      socket.close(); }; }

componentWillUnmount() {
    if (this.state.ws) {
        this.state.ws.close(); }}
```

REACT:<div style={myStyle}>(camel case)

```
import React from 'react';
import { api_getUsername, api_guess, api_newgame, api_getWins, api_getLosses }  from
'./api';
import { Header } from './header';
... MORE


class Main extends React.Component {
  constructor(props) {
    super(props);
    this.state = {
      stuff you want here
    };
    this.socket = new WebSocket(`ws://${window.location.hostname}:8151`);
    this.socket.onmessage=(Event => {console.log(Event.data)})
  }
  componentDidMount() {
    api_getUsername((data) => {
      this.setState({ username: data.username });
    });
  }
  handleNewGame = () => {
    this.setState({
      state you want to set
    });
    api_newgame(this.state.username);
  };

  handleIconClick = (name) => {
    this.setState({ activeComponent: name });
  };

   render() {
    let currComp;

    switch (this.state.activeComponent) {
      case 'ui_home':
        currComp = <Solo />;
        break;

            currComp = <Play state={this.state} guessCharacter={this.guessCharacter}
putCharacter={this.putCharacter} delCharacter={this.delCharacter}
handleNewGame={this.handleNewGame} />;
            break;

      case 'ui_stats':
        currComp = <Stats state={this.state}/>;
        break;

      default:
        currComp = null;
    }

    return (
      <div>
        <Header onIconClick={this.handleIconClick} />
        {currComp}
      </div>
    );
  }
}
export { Main };
class Header extends React.Component {
    state = {
      selectedPage: null
    };

    handleClick = (name) => {
      console.log(name);
      this.setState({ selectedPage: name });
      this.props.onIconClick(name);
    };

    render() {
      const { selectedPage } = this.state;

      return (
        html for nav
      );
    }
  }

export { Header };
```

JS-API

```
function api_getUsername(cb){
    let url="/api/username";
    fetch(url, {
                    method: "GET", // *GET, POST, PUT, DELETE, etc.
                    mode: "same-origin", // no-cors, *cors, same-origin
                    cache: "no-cache", // *default, no-cache, reload,
                    credentials: "same-origin", // include, *same-origin, omit
                    headers: {
                                "Content-Type": "application/json",},
                    redirect: "follow", // manual, *follow, error
                    referrerPolicy: "no-referrer", // no-referrer,
                    body: JSON.stringify(),
                    })
                    .then(response=>response.json())
                    .then(data=>cb(data))
                    .catch(error=>console.log(error));
}
```

Canvas: const canvas = document.getElementById("canvas");

```
const ctx = canvas.getContext("2d");
ctx.fillStyle = "green";
ctx.fillRect(10, 10, 150, 100);
```

Q3 b

```
<body>
<h1>LIST</h1>
<div id="list-container"></div>
<form id="add-item-form">
    <label>Add Item: <input type="text" id="new-item"></label>
    <button type="submit">Add</button>
</form>
</body>
</html>
```

3c

```
function getList() {
    $.ajax({
                    method: “GET”,
                    url: “/list”,
                    processData:false,
                    contentType: "application/json; charset=utf-8",
                    dataType:"json"
}).done(function(data,  text_status, jqXHR){this or html
                    console.log(jqXHR.status+" "+text_status+JSON.stringify(data));

}).fail(function(err){
        console.log("fail "+err.status+" "+JSON.stringify(err.responseJSON));
});
}

function addMore() {
    $.ajax({
                    method: “POST”,
                    url: “/list/more”,
      // i think should be ”/list/” + $(”#new-item”).val()
                    processData:false,
                    contentType: "application/json; charset=utf-8",
                    dataType:"json"
}).done(function(data,  text_status, jqXHR){
                    console.log(jqXHR.status+" "+text_status+JSON.stringify(data));

}).fail(function(err){
        console.log("fail "+err.status+" "+JSON.stringify(err.responseJSON));
});
}

// add event handlers?
$("#add-item-form button").on("click", (ev) => {
  ev.preventDefault(); // clicking button might reload page, so prevent that(?)
  addMore();
});
```

4a

```
// include code of previous application (maybe without routes)

const WebsocketServer = require('ws').Server;
const wss = new WebsocketServer({ port: port + 1 });

wss.on('connection', conn => {
  conn.on('message', (data, isBinary) => {
    const message = isBinary ? data.toString() : data; // mixed up in sampleServerSimple.js?
    const obj = JSON.parse(message);
    switch (obj.operation) {
      case 'add':
        myList.push(obj.data);
        return conn.send(JSON.stringify({ message: '' }));
      case 'delete':
        myList.splice(myList.indexOf(obj.data), 1);
        return conn.send(JSON.stringify({ message: '' }));
      case 'get':
        return conn.send(JSON.stringify({ list: myList }));
    }

  });
});
```

b)

untested solution code below

```
const conn = new WebSocket('ws://myserver.com:10001');

function getList() {
  conn.send(JSON.stringify({ operation: 'get' });
}
function addMore() {
  conn.send(JSON.stringify({ operation: 'add', $('#new-item').val() });
}
conn.addEventListener("message", ev => {
  const obj = JSON.parse(ev.data);
  if (obj.list) {
    // reconstruct table
    const table = $("<table><tr><th>Values</th></tr></table>");
    for (const item of obj.list) {
      table.append($("<tr><td>" + item + "</td></tr>")); // XSS injection vulnerable (but index.php
is also)
    }
    $('table').remove();
    $(document.body).append(table);
  }
});
```