AJAX (Asynchronous JavaScript + XML)

```javascript
function postData() {
  var xhttp = new XmlHttpRequest();
  xhttp.open("POST","page1",true);
  xhttp.send("var1=value1&var2=value2");
}

function getDataSend() {
  var xhttp = new XmlHttpRequest();
  xhttp.open("GET","/page2?var1=value1&var2=value2",true);
  xhttp.send();
}
```

Send Request to Server
```
xhttp.open(method,url,async);
         GET      true
         POST     false
```

Execute function when a request is received from the server 1 in response to a GET & POST request 1

```javascript
var xhttp = new XmlHttpRequest();
xhttp.onreadystatechange = function() {
  if(this.readyState == 4 && this.status == 200) {
    document.getElementById("demo").innerHTML = this.responseText;
  }
};
xhttp.open("GET","ajax_page",true);
xhttp.send();
```

SECTION
```
<section>text</section>
• content inside section is grouped appears
  as 1 entry in an outline on a page.
```

SEND GET REQUEST
```html
<script src="https://ajax.googleapis.com/ajax/libs/jquery/1.12.4/jquery.min.js"></script>
```
```javascript
document.getElementById("form").addEventListener("click",function(event){
  event.preventDefault();
  var req = new XmlHttpRequest();
  var url = "http://api.openweathermap.org/weather?zip=12345"
  req.open("GET",url,true);
  req.addEventListener("load", function() {
    if(req.status >=200 && req.status<400){
      var res = JSON.parse(req.responseText);
    }else{ /*error*/ }

  req.send(null);
});                          var object = {key: value};

req.open("POST",url,true);
req.setRequestHeader("Content-type","application/json");
req.addEventListener("load",function(){ if(req.status>=200 && <400){ if(req.status>=200&&<400)}//success
  JSON.parse(req.responseText)//>else & error}
});
req.send(JSON.stringify(object));
event.preventDefault();
```

HTML TEMPLATE
```html
<!doctype html>
<html>
<head>
  <meta charset="utf-8">
  <meta http-equiv=
    "X-UA-Compatible"
    content="IE=edge">
  <meta name="viewport"
    content="width=device-
      width,initial-
      scale=1">
  <title>title</title>
  <link rel="stylesheet"
    type="text/css"
    href="css/style.css">
</head>
<body>
  {{{body}}}
  <script src="js/script.js">
    </script>
</body>
</html>
```

FORM
```html
<form action="get" method="get">
  <fieldset>
    <legend>Some Get data</legend>
    <label>text</label>
    <label>name
      <input type="text" name="text
        maxlength=100 />
    </label></fieldset>
  <fieldset>
    <input type="radio" name="Candy"
      value="skittles">Skittles</input>
    <input type="submit" value="submit"/>
  </fieldset>
</fieldset>
```

TABLE
```html
<table>
  <caption>text</caption>
  <thead><tr>
    <th><th>col1/th>
    </th></thead>
  <tbody><tr><td>data
  </td></tr></tbody>
</table>
```

CSS TEMPLATE
```css
body {
  background-color: #fff;
}

.class .className {
  border-style: none/solid/dash
  border-color: #000;
}

#idName {
  ...
}

table tbody tr:nth-of-type(od
  { border-width: 2px; }

table tbody tr:nth-of-type(eve
  { border-width: 2px; }

table tbody td:nth-child(2) {
  color: #000; padding: 0;
  margin: 1px 2px 3px 4px;}

ul li:nth-child(3n+3) {
  /* select every 3rd item in
  unordered list. (i.e. 3,6,9,12*/

p:nth-of-type(z) {
  /* Selects every <P> element that
  is the 2nd P element of its parents
```

CSS Selectors
```
.class
#id
*            - selects all elements
element      - selects elem
div, p       - selects all div and
               P elements
div > p      - selects all P elements
               where the parent is
               a div element
[attribute]  - selects all elements
               with this attribute
[target=_blank]- selects elems w/
               target attribute set to
               _blank
a:active     - selects anchor elem
               w/ an active link
P:first-child - selects P elem
               that is the child first of its
               parent
P:nth-child(2) - selects P elem
               that is 2nd child of parent
P:nth-of-type(2) - selects
               2nd element of its parent
P:nth-last-child(2) - select
               P elems thats 2nd child counting
               from last child
```

```js
var foo = +0; //
//foo is a Number type
```

```js
function deepEqual(obj1, obj2) {
  if (typeof obj1 == "object" && obj1 != null &&
      typeof obj2 === "object" && obj2 != null)) {
    var obj1_length = Object.keys(obj1).length;
    var obj2_length = Object.keys(obj2).length;
    if (obj1.length === obj2.length) {
      for (var prop1 in obj1) {
        for (var prop2 in obj2) {
          if (prop1 == prop2) {
            if (!deepEqual(obj1[prop1], obj2[prop2]))
              return false;
          }
        }
      }
    }
  } else { return false; }
}

deepEqual([0,1,2], [0,1,3]); // false
var obj1 = {here: {is: "an", object: 2}}; var p =
deepEqual(obj1 {here:1, object: 2}); // false
(skoog)
```

JS CLASS
```
- functions are hoisted. JS classes are not.
  create class first, then access it.
class Rectangle {
  Constructor(height, width) {
    this.height = height;
    this.width = width;
  }
}
```

//HIGHER-ORDER //Functions
```js
function diglog(name) {
  return function(text) {
    return name + " says '" + text;
  };
}
var person = { name: "wife" };
person.talk = dialog(person.name);
console.log(person.talk("go take trash out!"));
```

```js
function Vector(x,y) {
  this.x = x;
  this.y = y;
  Object.defineProperty(this, "length", {
    get: function() {
      return Math.sqrt(Math.pow(this.x,2) +
        Math.pow(this.y,2));
    }})
}
Vector.prototype.Plus = function(obj) {
  return new Vector(this.x + obj.x, this.y + obj.y);
}
console.log(new Vector(1,2).plus(new Vector(2,3)));
```

POST SERVER-SIDE
```js
app.post('/', function(req, res, next) {
  var cmd = req.body['cmd'];
  switch(cmd) {
    case 'insert':
      var sql_var = [req.body['name'],
        req.body['reps'], etc];
      var str = "INSERT INTO workouts('name', reps,
        'weight', date, 'lbs') VALUES \
        (?,?,?,?,?)";
      pool.query(sql_str, sql_var, function(err, result) {
        if(err) { next(err); return; }
        res.end(JSON.stringify('result));
      });
  }
});
```

HOME.handlebars `<h1>web page</h1>`
```handlebars
<ul>{{#if data}}
<li>{{#each data}}
  {{this.key}}
  {{this.value}}
</li> {{each}}
{{if}} </ul>
```

```js
function showTable(res) {
  var sql_str = "SELECT * FROM 'workouts'"
  pool.query(sql_str, function(err, res) {
    if(err) { next(err); return; }
    res.render('name', {data: data});
  });
};
```

NODE.JS
```js
var express = require('express');
var app = express();
var handlebars = require('express-handlebars');
var bodyParser = require('body-parser');
var mysql = require('mysql');
app.use = (bodyParser.urlencode({extended: false}))
app.use(express.static('public');
var pool = mysql.createPool({
  connectionLimit: 10,
  host: 'classmysql.engr.oregonstate.edu',
  user: 'cs290_deae',
  password: 'pwd',
  database: 'cs290_deae',
});
module.exports.pool = pool;
app.engine('handlebars', require('express-handlebars').create({defaultLayout: 'main'}));
app.set('view engine', 'handlebars');
app.set('port', port);
```

```js
app.use(function(req, res, next) {
  res.status(404);
  res.render('404');
});

app.use(function(err, req, res, next) {
  console.error(err.stack);
  res.type('plain/text');
  res.status(500);
  res.render('500');
});

app.listen(app.get('port'), function() {
  console.log('Express started on ' + server + app.get('port')+'');
});
```

NODE.JS Variables
```
{{{var1}}} = HTML
{{var1}} = value without HTML
```

```js
app.get('/', function(req, res, next) {
  var sql_str = "SELECT 1 from information_schema.tables WHERE table_name=(?)"
  pool.query(sql_str, ['workouts'], function(err, results) {
    if(err) { next(err); return; }
    if(results.length ==0) {
      pool.query('CREATE TABLE workouts(
        id INT PRIMARY KEY AUTO_INCREMENT,
        name VARCHAR(255) NOT NULL,
        reps INT, date DATE, lbs BOOLEAN)" function(err) { if(err) next(err); return; }});
```