Server-Side Development

Lab 9

10/25

Project 3

- Due **tonight**
- **Office hours @ 2pm**

Project 4

- Last project
  - Project of your own choosing
  - May overlap with a personal project but may not overlap with any projects for other classes
  - Must use topics discussed in class but may run on any platform
    - client-side, server-side, iOS, dashboard widgets, Chrome extensions, etc.
- 3-5 project proposals due November 1st
- Decide on a project November 8th
- Project checkpoint November 15th
- Presentations November 29th & December 6th

Project 4 Proposals

- Turn in a set of 3-5 topics by 11/1
- Each description should be two paragraphs:
  - A paragraph describing the project
  - A paragraph describing (on a high level) how you plan on implementing it
- I will provide feedback and expect a finalized topic by 11/8

Project 4 Expectations

- Individual projects (except in very specific circumstances, email me).
- You should build an artifact that has a large interactive component.
  - Front end (i.e. interactive) code should have a significant amount of JavaScript.
  - Doesn't have to have a backend.
  - The artifact should have one or more of these: aesthetic value (i.e. interactive animation), entertainment value (i.e. a game), practical value.

Project 4 Expectations

- You should make substantial and interesting use of JavaScript code (if you just make numerous calls to different JavaScript libraries without any JavaScript logic you will get 0 points here).
- Again, the focus here is on building nice interfaces, so focus on first making a compelling interface (if you write a web app with a bad/boring interface you will not get full points).
- Your project should work, no mockups.
  - If your demo doesn't work then that's bad. Test your code!

Project 4 Example Ideas

- Take something you built from a previous project (i.e. a game for the drawing library or an interactive component from your state machines project) and build something really amazing from it.
- Build a cryptogram solver
  - Like http://www.blisstonia.com/software/WebDecrypto/ but make it look better.

Project 4 Example Ideas

- Build a game (using either the drawing library or something of your own).
- Build a 3D model that you can view (i.e. http://madebyevan.com/webgl-water/. Only attempt if you're good at graphics and have a lot of free time!)
- Roll your own image gallery viewer.
- Make your personal website much more compelling/ interactive (if you already had a website, save an old version of it so I can compare).

Project 4 Constraints

- You can use whatever JavaScript libraries you want, as long as you document it in your README for your submission.
- You should also mention libraries you used in your final presentation.
- REMINDER: piecing together libraries for this project will not give you full points under the ‘makes significant use of JavaScript’ criteria

Project 4 Submission

- E-mail all of the code needed for your project
- Include a README describing what you did, why it is interesting, how your work makes significant use of JavaScript, any problems you encountered or known bugs, and all of the libraries you used (with urls to references).

Project 4 Final Presentations

- Will be held November 29th & December 6th
- Each person will give a 5-10 minute demo of their project.
  - No powerpoint necessary, just show us what you built.
  - Talk about what was challenging about this project.
  - Show us at least one interesting feature of your project.

Project 4 Evaluation

- 100 points:
  - 10 points: turn in is correct (all code is included, README is present).
  - 10 points: formatting and comments.
  - 30 points: Substantial and interesting use of JavaScript code (if you just make numerous calls to different JavaScript libraries without any JavaScript logic you will get 10 points here).
  - 30 points: Aesthetic, entertainment, or practical value of what you made.
  - 20 points: Presentation (class will fill out score cards)

Last time…

- AJAX
- Server-side code
- Node.JS

Today

- Node.JS review
- Build an application using Node.JS
  - Real-time chat application
  - Work in teams of two (randomly selected)

what is "server-side?"

- most javascript is client-side
  - executed in the user's browser
- javascript may now be server-side
  - executed in the web page server
  - executed anywhere!

node.js

1. took Chrome javascript engine
2. made it an application
3. added libraries and utilities
4. result: javascript everywhere

http://node.js.org

node web server

```
var http = require('http');

http.createServer(function (req, res) {
  res.writeHead(200, {'Content-Type': 'text/plain'});
  res.end('Hello World');
}).listen(80);

console.log('Server running at http://127.0.0.1/');
```

node web server

```
var http = require('http');

http.createServer(function (req, res) {
  res.writeHead(200, {'Content-Type': 'text/plain'});
  res.end('Hello World');
}).listen(80);

console.log('Server running at http://127.0.0.1/');
```

node web server

```
var http = require('http');

http.createServer(function (req, res) {
  res.writeHead(200, {'Content-Type': 'text/plain'});
  res.end('Hello World');
}).listen(80);

console.log('Server running at http://127.0.0.1/');
```

node web server

```
var http = require('http');

http.createServer(function (req, res) {
  res.writeHead(200, {'Content-Type': 'text/plain'});
  res.end('Hello World');
}).listen(80);

console.log('Server running at http://127.0.0.1/');
```

node web server

```
var http = require('http');

http.createServer(function (req, res) {
  res.writeHead(200, {'Content-Type': 'text/plain'});
  res.end('Hello World');
}).listen(80);

console.log('Server running at http://127.0.0.1/');
```

node web server

```
var http = require('http');

http.createServer(function (req, res) {
  res.writeHead(200, {'Content-Type': 'text/plain'});
  res.end('Hello World');
}).listen(80);

console.log('Server running at http://127.0.0.1/');
```

node web server

```
var http = require('http');

http.createServer(function (req, res) {
  res.writeHead(200, {'Content-Type': 'text/plain'});
  res.end('Hello World');
}).listen(80);

console.log('Server running at http://127.0.0.1/');
```

node: socket.io

- create sockets for quick client-server communication
  - allows for very “live” web pages

```
socket.emit(event, data);

socket.on(event, function);
```

http://socket.io/

socket.io for node

server-side

```
var io = require('socket.io').listen(80);

io.sockets.on('connection', function (socket) {
  socket.emit('news', 'Stocks are up!');
  socket.on('response', function (data) {
    console.log(data);
  });
});
```

client-side

```
<script src="/socket.io/socket.io.js"></script>
<script>
  var socket = io.connect('http://localhost');
  socket.on('news', function (data) {
     socket.emit('response', { user: 'steve' });
  });
</script>
```

```
var io = require('socket.io').listen(80);

io.sockets.on('connection', function (socket) {
  socket.emit('news', 'Stocks are up!');
  socket.on('response', function (data) {
    console.log(data);
  });
});
```

client-side

```
<script src="/socket.io/socket.io.js"></script>
<script>
  var socket = io.connect('http://localhost');
  socket.on('news', function (data) {
     socket.emit('response', { user: 'steve' });
  });
</script>
```

```
var io = require('socket.io').listen(80);

io.sockets.on('connection', function (socket) {
  socket.emit('news', 'Stocks are up!');
  socket.on('response', function (data) {
    console.log(data);
  });
});
```

client-side

```
<script src="/socket.io/socket.io.js"></script>
<script>
  var socket = io.connect('http://localhost');
  socket.on('news', function (data) {
     socket.emit('response', { user: 'steve' });
  });
</script>
```

server-side

```
var io = require('socket.io').listen(80);

io.sockets.on('connection', function (socket) {
  socket.emit('news', 'Stocks are up!');
  socket.on('response', function (data) {
    console.log(data);
  });
});
```

```
<script src="/socket.io/socket.io.js"></script>
<script>
  var socket = io.connect('http://localhost');
  socket.on('news', function (data) {
     socket.emit('response', { user: 'steve' });
  });
</script>
```

server-side

```
var io = require('socket.io').listen(80);

io.sockets.on('connection', function (socket) {
  socket.emit('news', 'Stocks are up!');
  socket.on('response', function (data) {
    console.log(data);
  });
});
```

```
<script src="/socket.io/socket.io.js"></script>
<script>
  var socket = io.connect('http://localhost');
  socket.on('news', function (data) {
     socket.emit('response', { user: 'steve' });
  });
</script>
```

```
var io = require('socket.io').listen(80);

io.sockets.on('connection', function (socket) {
  socket.emit('news', 'Stocks are up!');
  socket.on('response', function (data) {
    console.log(data);
  });
});
```

client-side

```
<script src="/socket.io/socket.io.js"></script>
<script>
  var socket = io.connect('http://localhost');
  socket.on('news', function (data) {
     socket.emit('response', { user: 'steve' });
  });
</script>
```

server-side

```
var io = require('socket.io').listen(80);

io.sockets.on('connection', function (socket) {
  socket.emit('news', 'Stocks are up!');
  socket.on('response', function (data) {
    console.log(data);
  });
});
```

client-side

```
<script src="/socket.io/socket.io.js"></script>
<script>
  var socket = io.connect('http://localhost');
  socket.on('news', function (data) {
     socket.emit('response', { user: 'steve' });
  });
</script>
```

```
var io = require('socket.io').listen(80);

io.sockets.on('connection', function (socket) {
  socket.emit('news', 'Stocks are up!');
  socket.on('response', function (data) {
    console.log(data);
  });
});
```

client-side

```
<script src="/socket.io/socket.io.js"></script>
<script>
  var socket = io.connect('http://localhost');
  socket.on('news', function (data) {
     socket.emit('response', { user: 'steve' });
  });
</script>
```

server-side

```
var io = require('socket.io').listen(80);

io.sockets.on('connection', function (socket) {
  socket.emit('news', 'Stocks are up!');
  socket.on('response', function (data) {
    console.log(data);
  });
});
```

client-side

```
<script src="/socket.io/socket.io.js"></script>
<script>
  var socket = io.connect('http://localhost');
  socket.on('news', function (data) {
     socket.emit('response', { user: 'steve' });
  });
</script>
```

server-side

```
var io = require('socket.io').listen(80);

io.sockets.on('connection', function (socket) {
  socket.emit('news', 'Stocks are up!');
  socket.on('response', function (data) {
    console.log(data);
  });
});
```

client-side

```
<script src="/socket.io/socket.io.js"></script>
<script>
  var socket = io.connect('http://localhost');
  socket.on('news', function (data) {
     socket.emit('response', { user: 'steve' });
  });
</script>
```

```
socket.emit(event, data);

socket.on(event, function);

io.sockets.emit(event, data);
```

bonjour user_3

hello user_1

你好 user_5

¡Hola! user_2

http://from.so/web_lab/chat.html

Let’s implement a chat app!

- Install socket.IO. In your project folder, run:
  - `npm install socket.io`
- Create a file `chat_server.js`:

```
var app = require('http').createServer(handler)
    , io = require('socket.io').listen(app)
    , fs = require('fs')

app.listen(8000);

function handler (req, res) {
    fs.readFile(__dirname + '/index.html',
        function (err, data) {
            if (err) {
                res.writeHead(500);
                return res.end('Error loading index.html');
            }

            res.writeHead(200);
            res.end(data);
        });
}
```

Next time…

- CSS and styling
  - Will use CSS to add style to chat application