NEA Project Technical Solution

Jacob Halleron The Sandon School

Contents

Files

src/css/main.css

```
/*
    style sheet for the whole site
*/

body {
    font-family: monospace;
    font-size: 16px;
    background: #303030;
}
header {
    margin-bottom: 16px;
}
main {
    display: flex;
    flex-direction: column;
    align-items: center;
}
#charts-go-here {
    display: flex;
    flex-direction: row;
    flex-wrap: wrap;
    justify-content: center;
    align-content: center;
    align-items: center;
    height: 100%;
}
article {
    max-width: 800px;
    color: white;
}
figure {
    margin: 0px;
}
canvas {
    background: #303030;
}
h1 {
    font-size: 2em;
}
h2 {
    font-size: 1.5em;
}
h3 {
    font-size: 1.3em;
}
h4 {
    font-size: 1em;
    font-style: italic;
}
pre {
    width: 100%;
}
ul.variables {
    margin: 0px;
    padding: 0px;
}
ul.variables li {
    display: inline;
    list-style: none;
    margin-right: 1ch;
}
a {
    color: #7070FF;
}
select.algoSelect {
}
```

```
math {
    font-style: italic;
}
```

`src/html/404.html`

```
<!DOCTYPE html>
<html>
<head>
    <!-- 404 page for nonexistent requests -->
    <title>404</title>
    <link rel="stylesheet" type="text/css" href="/css/main.css"/>
</head>
<body>
    <main>
        <article>
            <h1>404 Error</h1>
            <p>Page not found.</p>
        </article>
    </main>
</body>
</html>
```

`src/html/index.html`

```
<!DOCTYPE html>
<html>
<head>
    <!-- homepage for requests with no path -->
    <title>NEA</title>
    <link rel="stylesheet" type="text/css" href="/css/main.css"></link>
</head>
<body>
    <main>
        <article>
            <h1>Jacob's NEA Project</h1>
            <p>Welcome to Jacob's A-level NEA project!</p>
            <ul>
                <li><a href="https://github.com/halogen487/visualiser">Repositor
                <li><a href="mailto:15b1halj@sandon.essex.sch.uk">Email</a></li>
                <li><a href="/single">Single animation test</a></li>
                <li><a href="/multiple">Multiple animation test</a></li>
            </ul>
        </article>
    </main>
```

```
</body>
</html>
```

`src/html/multiple.html`

```
<!DOCTYPE html>
<html>
<head>
    <!-- page for multiple charts -->
    <title>Visualiser</title>
    <link rel="stylesheet" type="text/css" href="/css/main.css"></link>
    <script src="/js/main.js" type="module" defer="yes"></script>
</head>
<body id="multiple">
    <header>
        <button class="control">play</button>
        <button class="control">pause</button>
        <button class="control">reset</button>
        <input type="number" class="speed" min="0" max="1000" placeholder="inter
        <input type="number" class="length" min="10" max="1000" placeholder="arr
    </header>
    <main>
            <div id="charts-go-here"></div>
            <button id="new-chart">add chart</button>
    </main>
</body>
</html>
```

`src/html/single.html`

```
<!DOCTYPE html>
<html>
<head>
    <!-- page for single chart -->
    <title>Visualiser</title>
    <!-- import script and stylesheet -->
    <link rel="stylesheet" type="text/css" href="/css/main.css"></link>
    <script src="/js/main.js" type="module" defer="yes"></script>
</head>
<body id="single">
    <header>
        <button class="control">play</button>
        <button class="control">pause</button>
        <button class="control">reset</button>
        <input type="number" class="speed" min="0" max="1000" placeholder="inter
        <input type="number" class="length" min="10" max="1000" placeholder="arr
    </header>
    <main>
            <div id="charts-go-here"></div>
    </main>
</body>
</html>
```

src/js/algos.js

```
export function Algorithm (init, step, chartType) { // algorithm class
    this.init = init
    this.step = step
    this.chartType = chartType
}

export var algos = {
    check: new Algorithm (
    // doesn't actually check, just does the whoosh
        function () {
            this.checki = 0;
            this.done = false;
        },
        function () {
            this.scanning = [this.checki, this.checki + 1];
            if (this.checki <= this.value.length) {
            } else {
              this.done = true;
            }
            this.checki++;
        },
        "sort"
    ),
    bogo: new Algorithm (
        function () {
            this.v.attempts = 0;
            this.ele.querySelector(".bigo").innerHTML = "O(n!)"
            try {this.ele.querySelector(".pseudocode").innerText = `
while not sorted:
    shuffle(array)
            `} catch {}
        },
        function () {
            this.v.attempts += 1;
            for (let i = this.value.length - 1; i > 0; i--) {
              let r = Math.floor(Math.random() * (this.value.length - 1));
```

```
                if (r) {
                }
                this.swap(i, r);
              }
              let goodArr = Array.from({ length: this.value.length }, (v, i) => i
              let done = true;
              for (let i in this.value) {
                if (this.value[i] != goodArr[i]) {
                  done = false;
                  break;
                }
              }
              if (done == true) {
                this.done = true;
              }
          },
          "sort"
      ),
      boggle: new Algorithm (
          function () {
              this.v.comparisons = 0;
              this.ele.querySelector(".bigo").innerHTML = "O(a^n)"
              try {this.ele.querySelector(".pseudocode").innerText = `
while not sorted:
    a = random index in array
    b = random index in array
    if (a > b) and (array[a] > array[b]):
        swap(a, b)
              `} catch {}
            },
          function () {
              let a = Math.floor(Math.random() * this.value.length);
              let b = Math.floor(Math.random() * this.value.length);
              this.scanning = [a, b];
              if (a > b) {
                let x = b;
                b = a;
                a = x;
              }
              this.v.comparisons++;
              if (this.value[a] > this.value[b]) {
                this.swap(a, b);
              }
              this.comparisons++;
              let goodArr = Array.from({ length: this.value.length }, (v, i) => i
              let done = true;
```

```
                for (let i in this.value) {
                    if (this.value[i] != goodArr[i]) {
                        done = false;
                        break;
                    }
                }
                if (done == true) {
                    this.done = true;
                }
            },
            "sort"
        ),
        bubble: new Algorithm (
            // clean up
            function () {
                this.v = {
                    n: this.value.length - 1,
                    newn: this.value.length - 1,
                    swapped: true,
                    comparisons: 0,
                };
                this.ele.querySelector(".bigo").innerHTML = "O(n^2)"
                this.scanning = [this.value.length, this.value.length + 1];
                try {this.ele.querySelector(".pseudocode").innerText = `
    n = length(array)
    repeat until n <= 1:
        newn = 0
        for i = 1 to n - 1:
            if array[i - 1] > array[i]:
                swap(i - 1, i)
                newn = i
        n = newn
                `} catch {}
            },
            function () {
                for (let i in this.scanning) {
                    this.scanning[i]++;
                }
                if (this.scanning[1] >= this.v.n) {
                    // at end
                    if (this.v.n <= 1) {
                        this.done = true;
                    } else {
                        this.v.swapped = false;
                        this.scanning = [0, 1];
                        this.v.n = this.v.newn + 1;
                        this.v.newn = 0;
                    }
                }
                let [a, b] = [this.scanning[0], this.scanning[1]]; // for easier rea
                this.v.comparisons++;
                if (this.value[a] > this.value[b]) {
                    this.swap(a, b);
                    this.v.newn = this.scanning[0];
                    this.v.swapped = true;
                }
            },
            "sort"
        ),
        cocktail: new Algorithm (
            function () {
                this.ele.querySelector(".bigo").innerHTML = "O(n^2)"
                /*try {this.ele.querySelector(".pseudocode").innerText = `

                `} catch {}*/
                this.v = {
                    swapped: true,
                    start: 0,
                    end: this.value.length,
                    up: true,
                }
                this.scanning = [this.value.length, this.value.length + 1]
            },
            function () {
                if (this.v.up) {
                    for (let i in this.scanning) {
                        this.scanning[i]++
                    }
                    if (this.scanning[0] >= this.v.end) {
                        this.v.up = false
                        this.v.end--;
                        console.log("down")
                        if (!this.v.swapped) {
                            this.done = true
                        }
                    }
                } else {
                    for (let i in this.scanning) {
                        this.scanning[i]--
                    }
                    if (this.scanning[0] <= this.v.start) {
                        this.v.up = true
                        this.v.start++;
                        this.v.swapped = false
                        console.log("up")
                    }
                }
                if (this.value[this.scanning[0]] > this.value[this.scanning[1]]) {
                    this.swap(this.scanning[0], this.scanning[1])
                    this.v.swapped = true
                }
            },
            "sort"
        ),
        insertion: new Algorithm (
            function () {
                this.v = {
                    i: 1,
                    j: 1,
                    comparisons: 0,
                }
                this.ele.querySelector(".bigo").innerHTML = "O(n^2)"
                try {this.ele.querySelector("pseudocode").innerText = ``} catch {}
            },
            function () {
                if (this.v.i < this.value.length) {
                    this.scanning = [this.v.j, this.v.j - 1];
                    this.v.comparisons++;
                    if (this.v.j > 0 && this.value[this.v.j - 1] > this.value[this.v
                        this.swap(this.v.j, this.v.j - 1);
                        this.v.j--;
                    } else {
                        this.v.i++;
                        this.v.j = this.v.i;
                    }
                } else {
                    this.done = true;
                }
            },
            "sort"
        ),
        selection: new Algorithm (
            function () {
                this.v = {
                    jMin: 0
                }
                this.scanning = [0, 1]
            },
            function () {
                if (this.scanning[0] < this.value.length - 1) {
                    if (this.scanning[1] < this.value.length) {
                        if (this.value[this.scanning[1]] < this.value[this.v.jMin])
                            this.v.jMin = this.scanning[1]
                        }
                        this.scanning[1]++
                    } else {
                        this.scanning[1] = this.scanning[0] + 1
                        if (this.v.jMin != this.scanning[0]) {
                            this.swap(this.scanning[0], this.v.jMin)
                        }
                        this.scanning[0]++
                        this.v.jMin = this.scanning[0]
                    }
                } else {
                    this.done = true
                }
            },
            "sort"
        )
    }
```

src/js/graphjunk.js

```
function GraphNode (id, to) {
    this.id = id
    this.to = to
    this.from = []
    this.x = null
    this.y = null
}

function Graph () {
    this.nodes = []
    this.edges = []
}

function GraphChart (nodeCount, maxTos) {

    Chart.call(this)

    this.oldReset = function () {
        this.value = []
        // init nodes
        for (let i = 0; i < nodeCount; i++) {
            let toCount = Math.floor(Math.random() * maxTos) + 1
            let tos = []
            for (let j = 0; j < toCount; j++){tos.push(Math.floor(Math.random()
            this.value.push(new GraphNode(i, tos))
        }
        // assign froms
        for (let i of this.value) {
            for (let j of this.value) {
                if (j.to.indexOf(i.id) >= 0) {
                    i.from.push(j.id)
                }
            }
        }
        console.log(this.value)
        return this
    }

    this.reset = function () {
        this.value = []
    }

    this.draw = function () {

    }

    this.running = null
    this.reset()
}
```

src/js/main.js

```
import {algos} from "/js/algos.js"

/*
    test script for charts
*/

console.info("rectangle.js is alive")

function Chart () { // class for any chart
    if (charts[0]) { // give self unique chart ID
        this.id = Number(Object.keys(charts)[Object.keys(charts).length - 1]) +
    } else {
        this.id = 0
    }
    document.querySelector("#charts-go-here").insertAdjacentHTML("beforeend", `
    <article id="chart${this.id}">
        <header>
            <select class="algoSelect">
                <option disabled selected>select algorithm</option>
            </select>
            <math class="bigo">O(n)</math>
        </header>
        <figure>
            <canvas width="${config.single ? 784 : 576}" height="${config.si
            <figcaption>
                <ul class="variables"></ul>
            </figcaption>
        </figure>
    ${config.single ? `
            <!--pre><code class="pseudocode"></code></pre--> <!-- only show
        ` : ``}
    </article>
`)
this.ele = document.querySelector(`#chart${this.id}`) // element object for
this.ctx = this.ele.querySelector("canvas").getContext("2d") // canvas conte
this.ele.querySelector(".algoSelect").addEventListener("change", (evt) => {
    charts[this.id].setAlgo(evt.target.value)
})
this.pause = function () { // method to stop running
    clearInterval(this.running)
    this.running = null
    this.vol.gain.value = 0
    return this // most methods return themselves so you can run multiple me
}

this.setAlgo = function (algo) { // method to change algorithm
    this.algo = algo
    if (algo != "check") {this.actualAlgo = algo}
    this.scanning = []
    if (this.algo) {
        if (algo != "check") {
            this.v = {}
        }
        algos[this.algo]["init"].apply(this)
    }
    this.draw()
    return this
}
this.setSpeed = function (ms) { // method to change speed
    this.pause()
    this.interval = ms
```

```
        if (this.running) {
            this.play()
        }
        return this
    }

    this.running = null // when running this contains the setInterval() loop ID
    this.value = null // array that the chart represents
    this.shownValue = null // array currently being shown
    this.algo = null
    this.interval = 50
    this.scanning = null
}

function SortChart (length) { // class for a chart that shows a sorting algorithm

    Chart.call(this) // subclass of Chart, I was originally going to have multip

    this.done = false // true if the algorithm finished running
    this.swap = function (a, b) { // swaps values at given array indices
        let t = this.value[a] // temporary variable
        this.value[a] = this.value[b]
        this.value[b] = t
        return this
    }

    this.draw = function () { // method to draw the chart itself to the appropri
        if (this.running && this.scanning[0] && config.sound) {this.beep(this.va
        // calculate changes between shown array and real array
        let moves = []
        for (let i in this.shownValue) {
            moves.push(this.value.indexOf(this.shownValue[i]))
        }
        // write algorithm name and variables
        this.ele.querySelector(".variables").innerHTML = ""
        for (let i in this.v) {
            this.ele.querySelector(".variables").innerHTML += `<li>${i}: ${this.
        }
        this.ctx.clearRect(0, 0, this.ctx.canvas.width, this.ctx.canvas.height)
        let barWidth = this.ctx.canvas.getAttribute("width") / this.value.length
        let rectHeight = Number(this.ctx.canvas.getAttribute("height"))
        for (let i in this.value) {
            // draw bar
            let barUnit = rectHeight / Math.max.apply(null, this.value) // heigh
            this.ctx.fillStyle = "#f7f7f7"
            if (moves[i] != i) { // if bar moved, make it fully white
```

```
                this.ctx.fillStyle = "#ffffff"
            }
            if (this.algo == "check" && i < this.checki) { // turns green at end
                this.ctx.fillStyle = "lime"
            }
            if (
                this.scanning.indexOf(Number(i)) >= 0 // if bar is being "scanne
                && this.running
            ) {
                this.ctx.fillStyle = "red"
            }
            this.ctx.fillRect(i * barWidth, rectHeight - (barUnit * this.value[i
        }
        // reset shownValue
        this.shownValue = []
        for (let i of this.value) {this.shownValue.push(i)}
        return this
    }

    this.play = function () { // method to start running
        console.info("chart", this.id, "playing", this.algo)
        if (!this.running && this.algo) {
            if (this.done) {
                this.reset()
            }
            if (this.algo) {
                this.running = setInterval(() => { // setInterval runs this func
                    algos[this.algo]["step"].apply(this)
                    this.draw()
                    if (this.done && this.algo != "check") { // if done, run che
                        this.setAlgo("check")
                        this.play()
                    } else if (this.done) { // if it's already run check algorith
                        this.pause()
                    }
                }, this.interval)
            }
            try {
                this.oscillator.start() // try to start beeping, won't work if i
            } catch {}
        }
        return this
    }
    this.setLength = function (n) { // method to change size of the array chart
        let o = this.value.length
        this.value.splice(o - (o - n))
        for (let i = 0; i < n - o; i++) {
            this.value.push(o + i + 1)
        }
        this.draw()
        return this
    }
    this.reset = function (length) { // shuffle
        if (!length) {length = this.value.length}
        this.pause()
        if (this.algo == ”check”) {this.algo = this.actualAlgo}
        this.value = Array.from({length: length}, (n, i) => i + 1)
        for (let i = length - 1; i > 0; i--) {
            let r = Math.floor(Math.random() * (length - 1))
            this.swap(i, r)
        }
        this.setAlgo(this.algo)
        this.scanning = []
        this.done = false
        if (this.algo) {algos[this.algo][”init”].apply(this)}
        this.draw()
        return this
    }
    this.beep = function (height) { // makes one beep
        try {
            this.vol.gain.value = 0.5
            this.oscillator.frequency.value = height + 300
        } catch {}
        return this
    }

    for (let i of Object.keys(algos).filter((key) => {return (algos[key].chartTy
        this.ele.querySelector(”.algoSelect”).insertAdjacentHTML(”beforeend”, ‘
            <option value=”${i}”>${i}</option>
        ‘)
    }

    this.running = null
    this.actualAlgo = null
    this.actx = new (window.AudioContext || window.webkitAudioContext)()
    this.oscillator = this.actx.createOscillator()
    this.vol = this.actx.createGain()
    this.oscillator.connect(this.vol)
    this.vol.connect(this.actx.destination)
    this.vol.gain.value = 0
    this.oscillator.type = ”sine”
```

```
    this.reset(length)
    this.draw()
}

function buttHandler (evt) { // handles any button push, runs function depending
    for (let chart in Object.keys(charts)) {
        charts[chart][evt.target.innerText]()
    }
}

function addChart (chart) { // function to add new chart
    charts[chart.id] = chart
}

function removeChart (id) { // function to remove a chart
    delete charts[chart.id]
}

var config = { // configuration, user can't access unless they somehow edit this
    loop: false,
    sound: true,
    pageType: document.querySelector("body").getAttribute("id")
}
if (config.pageType == "single") {config.single = true}

var charts = {} // map of all chart objects

for (let i of document.querySelectorAll(".control")) { // attach button handler
    i.addEventListener("click", buttHandler)
}
document.querySelector(".speed").addEventListener("change", (evt) => { // attach
    for (let i in Object.keys(charts)) {
        charts[i].setSpeed(evt.target.value)
        if (charts[i].running) {
            charts[i].play()
        }
    }
})
document.querySelector(".length").addEventListener("change", (evt) => { // attac
    for (let i in Object.keys(charts)) {
        charts[i].setLength(evt.target.value)
        charts[i].reset()
    }
})
try { // attach addChart function to new chart button click event
    document.querySelector("#new-chart").addEventListener("click", () => {
        addChart(new SortChart(40))
    })
} catch {}

if (config.single) { // if on single animation page, add one chart (user won't b
    addChart(new SortChart(40))
}
```

src/js/treejunk.js

```
function TreeNode (id, to) {
    GraphNode.call(this)
}

function TreeChart (height, maxChildren) {

    this.reset = function () {
        let childCount = Math.floor(Math.random() * maxChildren + 1)
        let children = []
        if (height > 1) {
            for (let i = 0; i < childCount; i++) {
                children.push(this.generateTree(height - 1, maxChildren))
                this.r++
            }
        }
        return this
    }

    this.draw = function () {

        function calculateInitialX (tree) {
            for (child of tree.children) {
                calculateInitialX(child)
            }
            if (tree.children.length == 0) {
                if (tree) {}
            }
        }

        calculateInitialX(this.tree)

    }

    // returns array of IDs for the given traversal
    this.traverse = function (order) {
        function pre (node) {
            let trav = [TreeNode.id]
            for (i of node.to) {
                trav.concat(this.tree[i])
            }
        }

        function post (TreeNode) {

        }

        let traversal = []
        if (order == "pre") {traversal = pre()}
        for (i of this.to) {traversal = traversal.concat(i.traverse(order))}
        if (order == "post") {traversal.push(this.id)}
        return traversal
    }

    this.r = 1
    this.height = height
    this.maxChildren = maxChildren

    this.reset()
}
```

src/app.js

```
#!/usr/bin/env node

/*
    core node.js server
*/

const express = require("express"); const app = express() // import express, req
const path = require("path") // import file path manipulation functions
const fs = require("fs") // import filesystem control functions

app.use(function (req, res, next) { // log all HTTP requests to console and log
    let line = `${(new Date).toISOString()}: ${req.method} request for ${req.url
    console.log(line)
    fs.appendFile(path.join(__dirname, "http.log"), line + "\n", (err) => {if (e
    next()
})

app.use("/css/:cssId", function (req, res) { // requests for /css/... go to css
    res.sendFile(path.join(__dirname, "css", req.params.cssId))
})
app.use("/js/:jsId", function (req, res) { // requests for /js/... go to js fold
    res.sendFile(path.join(__dirname, "js", req.params.jsId))
})

app.get("/", function (req, res) { // request for homepage
    res.sendFile(path.join(__dirname, "/html/index.html"))
})
app.get("/single", function (req, res) { // request for single animation page
    res.sendFile(path.join(__dirname, "/html/single.html"))
})
app.get("/multiple", function (req, res) { // request for multiple animation pag
    res.sendFile(path.join(__dirname, "/html/multiple.html"))
})
app.use("/", function (req, res) { // request for anything else returns 404 page
    res.sendFile(path.join(__dirname, "/html/404.html"))
})

app.listen(80, () => { // listen on port 80
    console.info("server alive")
})
```

src/http.log

```
This is the log file for any HTTP requests.
2022-01-10T11:18:03.569Z: GET request for / from ::ffff:127.0.0.1
2022-01-10T11:18:03.683Z: GET request for /css/main.css from ::ffff:127.0.0.1
2022-01-10T11:18:14.792Z: GET request for /multiple from ::ffff:127.0.0.1
```

There's more to this file but it gets very repetitive.

README.md

```
# visualiser

This is Jacob's A-level NEA project. It's a web server, run start.sh as root and
```

start.sh

```
# simple bash startup script

sudo systemctl stop apache2
sudo ./src/app.js
```