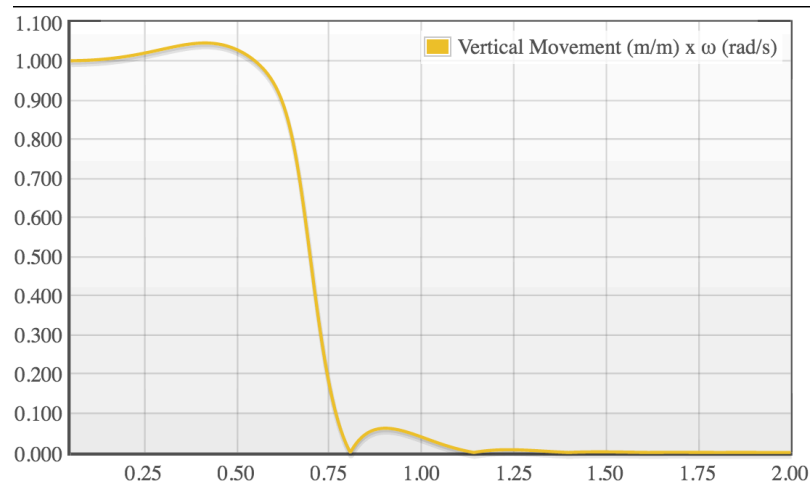
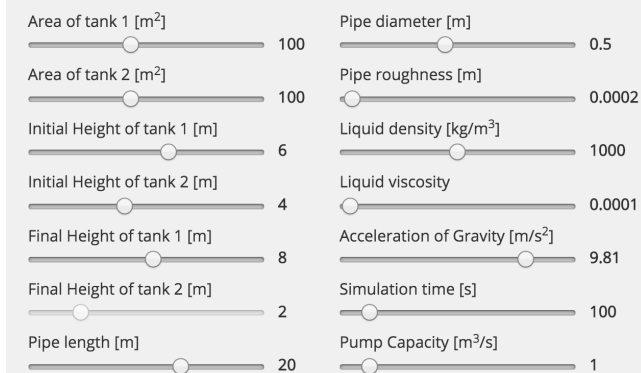


Introduction to JavaScript Applied to Design and Engineering

an informal talk at University College London (UCL)
April 21st 2016, London - UK

Parameters for user input



Assoc. Prof. Henrique M. Gaspar, PhD
Faculty of Maritime Technology and Operations - NTNU
henrique.gaspar@ntnu.no

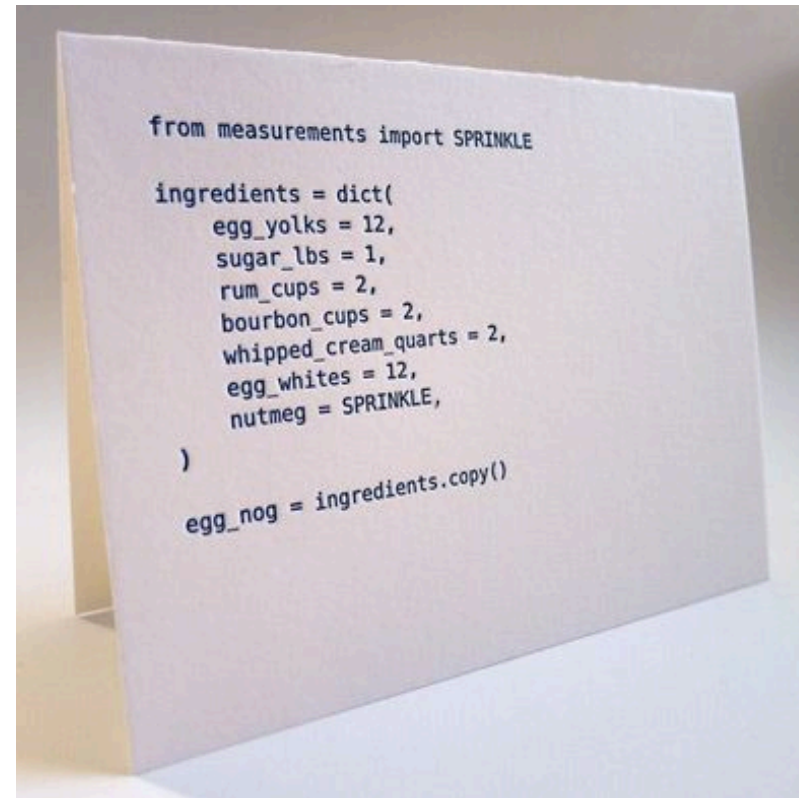
I – INTRODUCTION TO JS

Script (ing) Language

- Programming language that supports scripts, programs written for a special run-time environment that automate the execution of tasks that could alternatively be executed one-by-one by a human operator.
- Often interpreted (rather than compiled)
- Typically use abstraction, a form of information hiding, to spare users the details of internal variable types, data storage, and memory management.
- `a = 2; b = 3; a + b = 5`
- `a = "Hello "; b = "World"; a+b = "Hello World";`

Examples

- Matlab
- Mathematica
- Python
- C#
- Ruby
- **Javascript ***
- ...
- **https://en.wikipedia.org/wiki/List_of_programming_languages_by_type#Scripting_languages**



* Javascript IS NOT Java

Javascript

- One of the core languages of the web (+ CSS, HTML)
- 20 years old
- Made originally to control dynamically webpages
- Last 10 years developments (WebApps, Smartphones) spread the use to more advance apps
- Big companies (e.g Google) developed in the last years huge improvements to script parsing and interpretation
- Most of the cases close to C and Java nowadays, usually faster than Python, Matlab and C#
- Extremely easy to share (browser as GUI + Compiler + Console)

* Javascript IS NOT Java

When not Javascript?

When not to use Javascript:

- If Excel can solve your problem in short time
- When you don't need to use VBA in Excel
- When you don't need share your results
- When you don't need to interact with your model/ simulation parameters and/or data
- When you don't require any GUI
- When you want to control memory/events access
- When it can be solved without coding/computer (e.g sketching by hand/board)

Why Javascript? #1 Speed

<http://julialang.org/>



Why Javascript? #1 Speed

- Fastest Script Language (2015)
- Up to 120x faster than Matlab
- Up to 20x faster than Python
- Pretty good compared to C, Java and Fortran

| | Fortran | Julia | Python | R | Matlab | Octave | Mathe- matica | JavaScript | Go | LuaJIT | Java |
|---------------|-----------|-------|--------|--------|--------|---------|------------------|------------------|-------|--------------------|----------|
| | gcc 5.1.1 | 0.4.0 | 3.4.3 | 3.2.2 | R2015b | 4.0.0 | 10.2.0 | V8 3.28.71.19 | go1.5 | gsl-shell 2.3.1 | 1.8.0_45 |
| fib | 0.70 | 2.11 | 77.76 | 533.52 | 26.89 | 9324.35 | 118.53 | 3.36 | 1.86 | 1.71 | 1.21 |
| parse_int | 5.05 | 1.45 | 17.02 | 45.73 | 802.52 | 9581.44 | 15.02 | 6.06 | 1.20 | 5.77 | 3.35 |
| quicksort | 1.31 | 1.15 | 32.89 | 264.54 | 4.92 | 1866.01 | 43.23 | 2.70 | 1.29 | 2.03 | 2.60 |
| mandel | 0.81 | 0.79 | 15.32 | 53.16 | 7.58 | 451.81 | 5.13 | 0.66 | 1.11 | 0.67 | 1.35 |
| pi_sum | 1.00 | 1.00 | 21.99 | 9.56 | 1.00 | 299.31 | 1.69 | 1.01 | 1.00 | 1.00 | 1.00 |
| rand_mat_stat | 1.45 | 1.66 | 17.93 | 14.56 | 14.52 | 30.93 | 5.95 | 2.30 | 2.96 | 3.27 | 3.92 |
| rand_mat_mul | 3.48 | 1.02 | 1.14 | 1.57 | 1.12 | 1.12 | 1.30 | 15.07 | 1.42 | 1.16 | 2.36 |

Figure: benchmark times relative to C (smaller is better, C performance = 1.0).

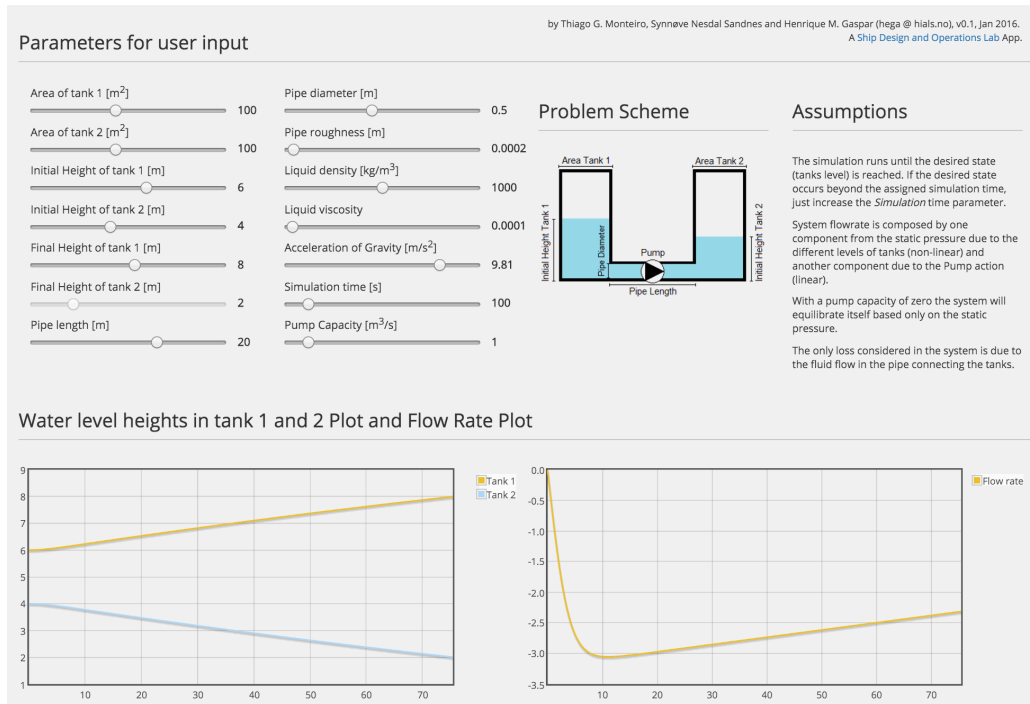
<http://julialang.org/>

* Javascript IS NOT Java

Why Javascript? #2 Compatibility

- It passes the “Shell Test”

“I think the Shell test is an imaginary person at Shell that wants to explore data/output on their desktop without having to install anything”



Traut and Smith, 2016

Why Javascript? #2 Compatibility


- Runs direct from any modern browser
- No need to install anything
- Can be shared online (server) and private (.HTML file)
- Requires no explanation when proper GUI
- No run/compiler
- Compatibility with PC, Smartphone, Tablets, OS



Try on your phone:

<http://www.shiplab.hials.org/app/3dconfigurator/>

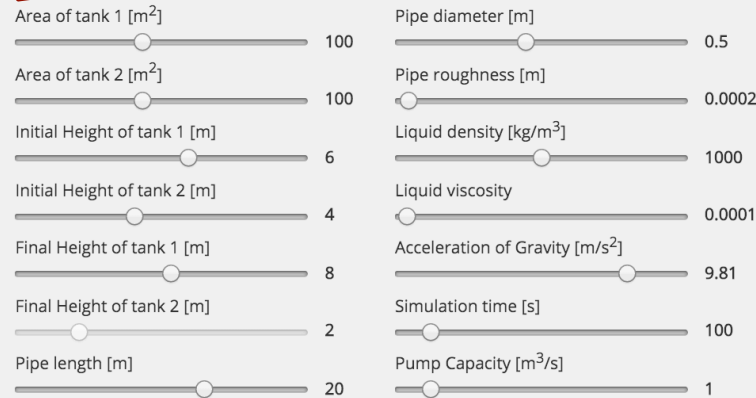
Why Javascript? #3 User Interface

- Browser as main user interface – HTML + CSS
- Everyone knows how to use a browser
- Difficult to create GUI in other languages (Python, Matlab, C#)
- Ready made elements in JS `<button>` 
- Every element of the page is customizable, clickable, editable
- From simple “click to run” to advanced 3D animation
- Sliders, Canvas, SVG
- Every page is a GUI example, from NYT to Wolfram

Why Javascript? #3 User Interface

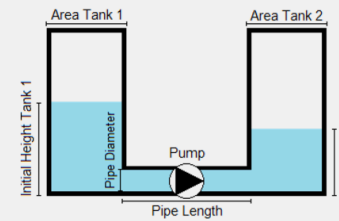
user input

Parameters for user input



by Thiago G. Monteiro, Synnøve Nesdal Sandnes and Henrique M. Gaspar (hega@hials.no), v0.1, Jan 2016.
A Ship Design and Operations Lab App.

Problem Scheme



text

Assumptions

The simulation runs until the desired state (tanks level) is reached. If the desired state occurs beyond the assigned simulation time, just increase the *Simulation* time parameter.

System flowrate is composed by one component from the static pressure due to the different levels of tanks (non-linear) and another component due to the Pump action (linear).

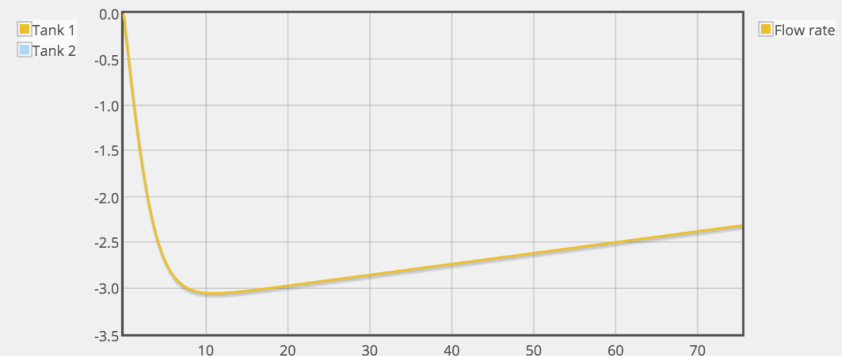
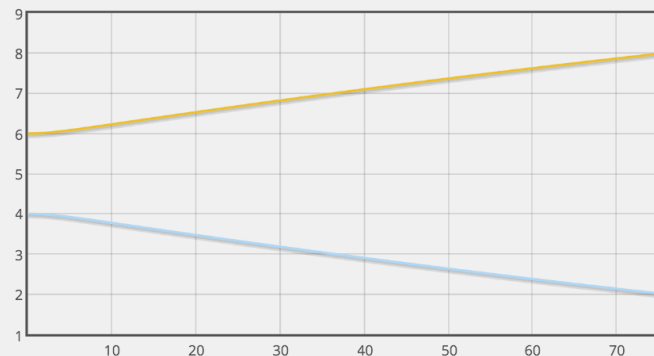
With a pump capacity of zero the system will equilibrate itself based only on the static pressure.

The only loss considered in the system is due to the fluid flow in the pipe connecting the tanks.

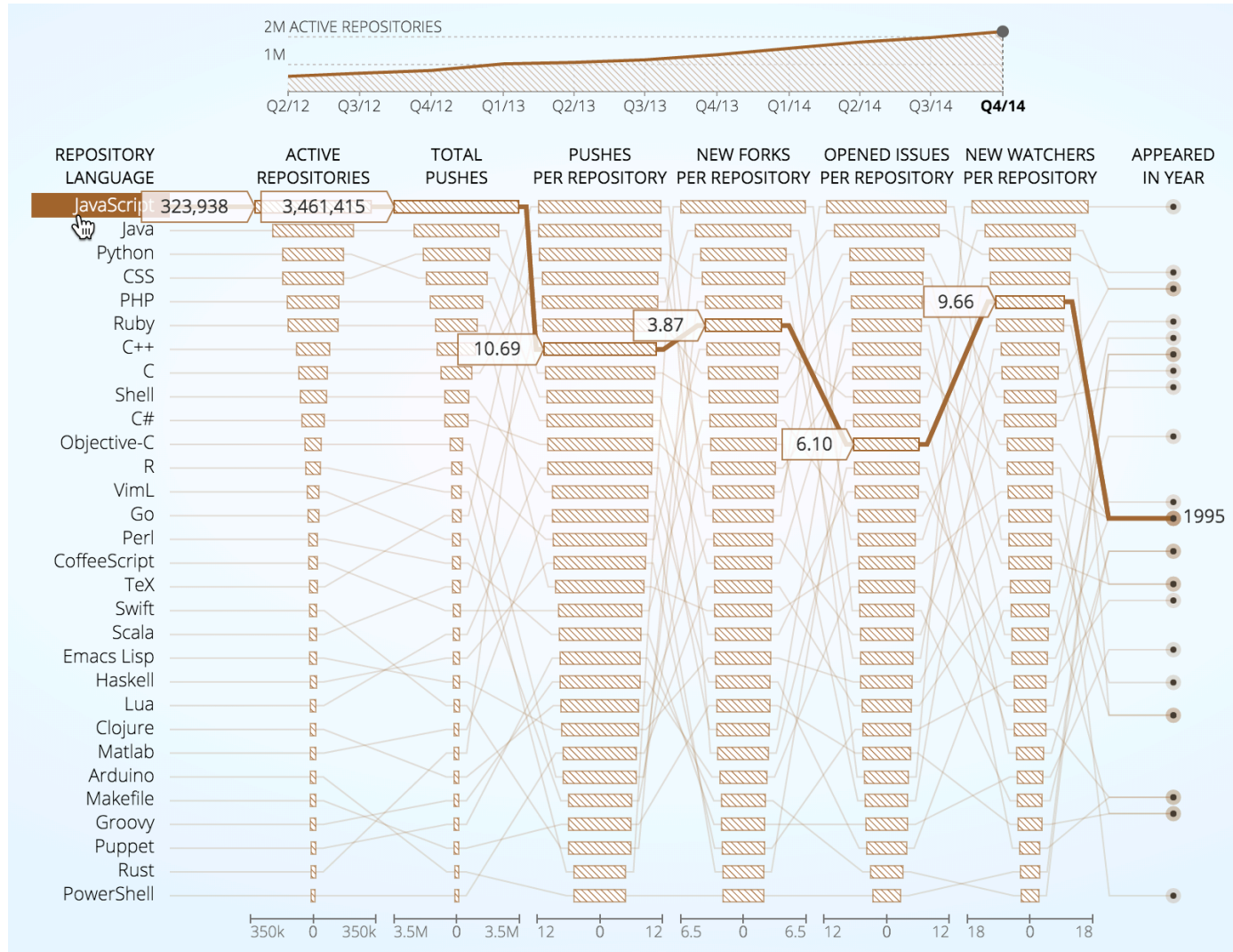
images

plots

Water level heights in tank 1 and 2 Plot and Flow Rate Plot

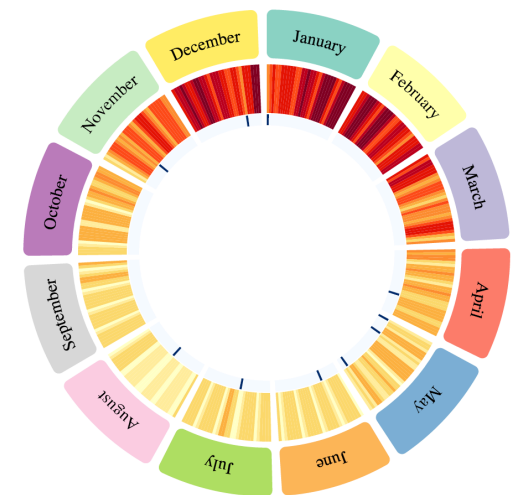


Why Javascript? #4 Usage



Why Javascript? #4 Usage

- Most common language among developers
- Free and open source
- Backed by the big guys in the industry (e.g. Google)
- Large amount of info available online
- Wide range of pre-made libraries:
 - Numeric.js (numerical calculation, matrices)
 - WebGL (3D graphs)
 - D3 (data handling methods and plots)
 - Snap (2D graphs)
 - Plotly and Flot (plots)
 - Node.js (server)
 - Processing (animation)



Why Javascript? #5 It works

- Most engineering calculations are developed within Excel
- Matlab has the market share when programming
- Javascript just got “excellence” few years ago, hugely helped by HTML5 simplicity
- Today, it just works
- Experience last 2 years – same code (even if bad code)
- Many ways to declare data (number, vector, matrices, object)
- Object and Prototypes
- Constrained by imagination, not lack of features

II – BASICS

Intro to JS – Simple Barge

- Console
- Editor (brackets, notepad ++)
- Loops (for, if)
- Objects and Prototypes
- Playing with the document (HTML)
- A button to run
- External libraries (D3, nvd3)
- A plot to appear
- Handling multivariate data
- A slider to change
- SVG and advanced thinking
- Case Studies

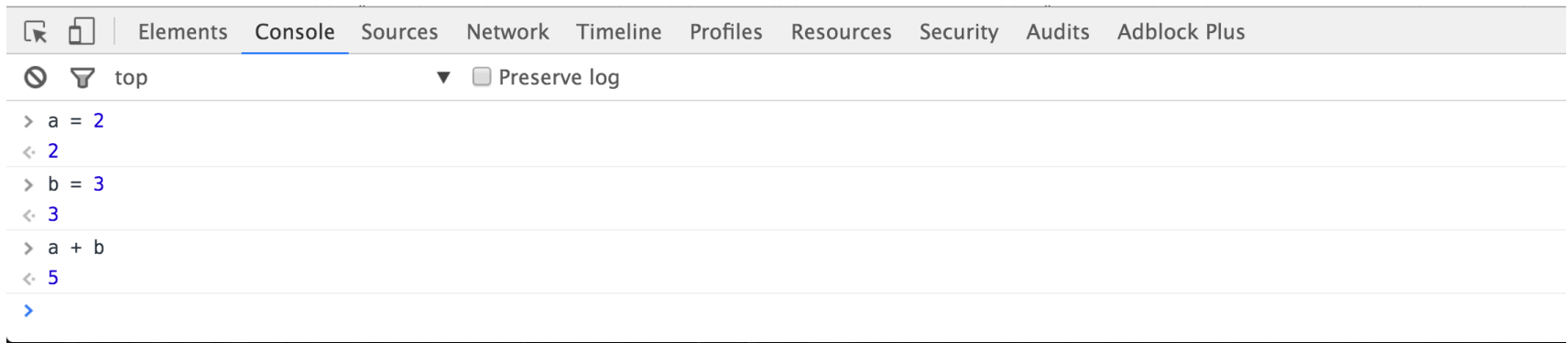
Intro to JS - Environment

- Your browser is already apt to run JS
- Developer tools

[Numeric.js 1.2.6](#)

Very quick new release. Davy Wybiral found a bug with `numeric.diveq([[1,2],[3,4]],2)` and in fixing it, I realized that some of the optimizations in Numeric were not optimal for the modern V8 JIT (I think something must have changed in the last 12 months). This resulted in a significant rewrite of the vector functions such as `numeric.add()`. On my computer, this resulted in a 33% speedup of the [benchmark](#). The bug that Davy found has also been fixed.

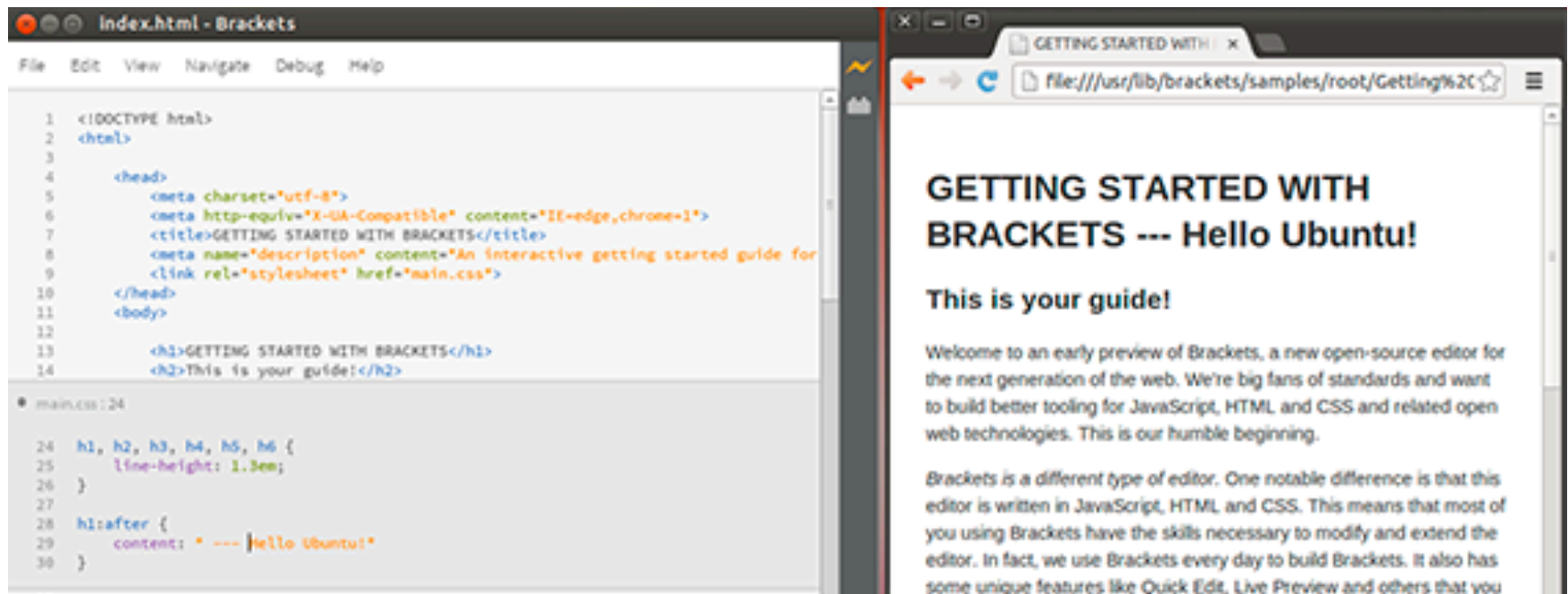
The new version of the `numeric.xxseq()` functions (eg. `numeric.addeq()`) can also now operate on typed Arrays, although I did not see significant performance benefits when I tested it. If you use `numeric.addeq()` on several different typed Array types, the JIT will realize that



* Use Chrome, always

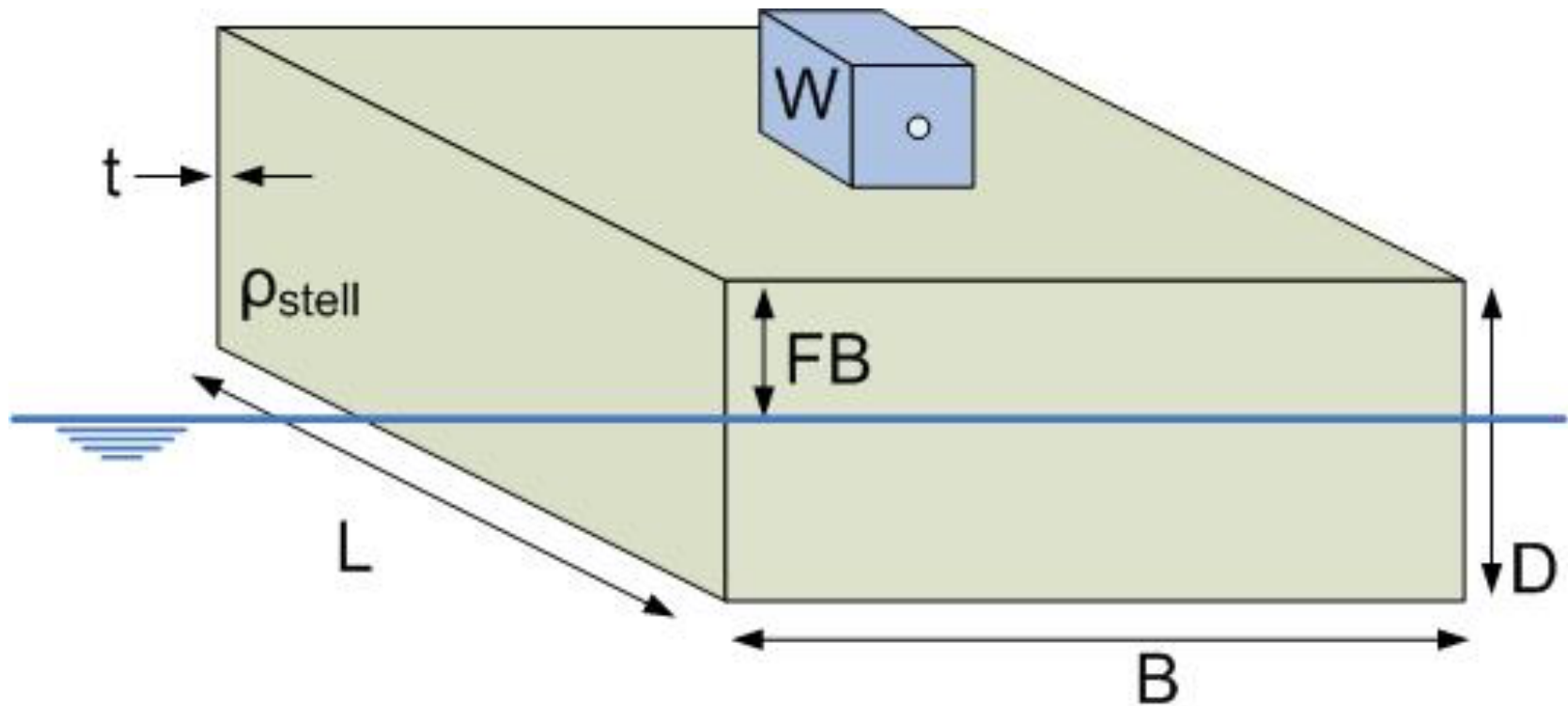
Intro to JS - Code

- Similar function and syntax as other languages
- For, If, While
- Use a good external editor (Brackets.io)



Intro to JS – Console

- Barge Problem



Intro to JS – Console

- Barge Problem

```
> L = 100
  B = 40
  D = 20
  lwt = 0.3*L*B*D
  ro = 1.025
  Tlwt = lwt/(L*B*1.025)
  W = 5000
  T = Tlwt + W/(L*B*1.025)
< 7.073170731707317
```

Intro to JS – Editor

- Brackets.io

```
1  L = 100
2  B = 40
3  D = 20
4  lwt = 0.3*L*B*D
5  ro = 1.025
6  Tlwt = lwt/(L*B*1.025)
7  W = 5000
8  T = Tlwt + W/(L*B*1.025)
```

Intro to JS – HTML

- JS included in HTML (browser readable)
- `<script></script>`
- As if in Matlab
- Not so intuitive (yet) as excel

```
<html>|
<script>
//Comment
// ; good practice when ending the line
L = 100;
B = 40;
D = 20;
lwt = 0.3*L*B*D;
ro = 1.025;
Tlwt = lwt/(L*B*1.025);
W = 5000;
T = Tlwt + W/(L*B*1.025);

//output
console.log('T = ', T);
document.write('T = ', T);
</script>
</html>
```

Intro to JS – HTML

- All in HTML is identifiable
- We can “get” and replace, add, etc.
- `<div id="">`
- ``
- Use HTML to make it beautiful

```
<html>
<body>
T = <span id="result"></span>
</body>
<script>
//Comment
// ; good practice when ending the line
L = 100;
B = 40;
D = 20;
lwt = 0.3*L*B*D;
ro = 1.025;
Tlwt = lwt/(L*B*1.025);
W = 5000;
T = Tlwt + W/(L*B*1.025);
|
//output
console.log('T = ', T);
document.getElementById('result').innerHTML = ('T = ', T);
</script>
</html>
```

Intro to JS – function

- Function needs to be loaded and called
- Order of arguments is important

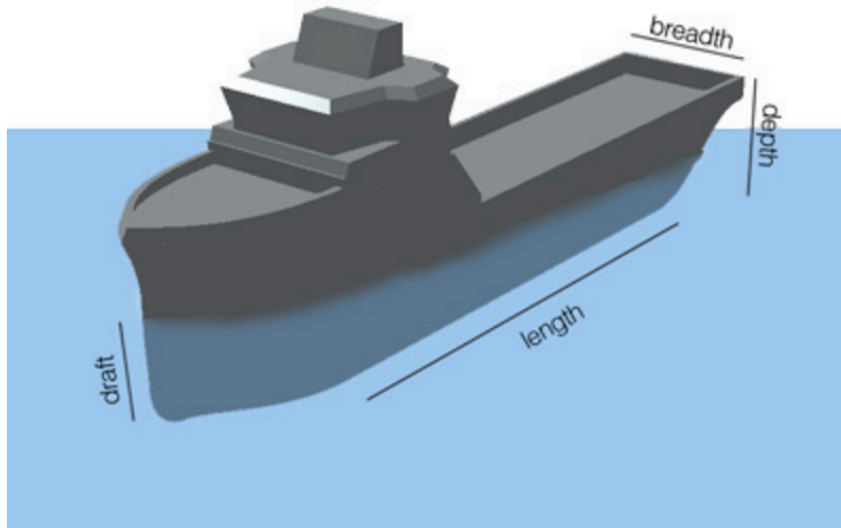
```
<script>
//Comment
// ; good practice when ending the line
L = 100;
B = 40;
D = 20;
lwt = 0.3*L*B*D;
ro = 1.025;
W = 5000;
[T,FB] = calc_draft_fb(L,B,D,lwt,W,ro)

function calc_draft_fb(l,b,d,lwt,w,ro){
  t = (lwt+w)/(l*b*ro);
  fb = d-t;
  return [t,fb];
}

//output
document.getElementById('result').innerHTML
L = ('T = ' + T + ', FB = ' + FB);
</script>
```

Intro to JS – Ship as object

object



properties and methods

```
// Properties
ship.Name = Ulstein
ship.Length = 100
ship.Breadth = 20
ship.Depth = 10
ship.Draft = 7
ship.Volume_Submerged = 5100
ship.Displacement = 5000
ship.Payload = 4000

//Methods
ship.Sail()
ship.Idle()
ship.Anchoring()
ship.Dynamic_Positioning()
ship.Anchor_Handling()
ship.Crane_Operation()
```

Intro to JS – Ship as object

- Object-oriented
- {Barge.property}
- It can be anything (number, function, string, object)

```
<script>
//Ship as object
Barge = {};

Barge.L = 100;
Barge.B = 40;
Barge.D = 20;
Barge.lwt = 0.3*Barge.L*Barge.B*Barge.D;
ro = 1.025;
W = 5000;
[Barge.T,Barge.FB] =
calc_draft_fb(Barge,W,ro)

function calc_draft_fb(ship,w,ro){
t = (ship.lwt+w)/(ship.L*ship.B*ro);
fb = ship.D-t;
return [t,fb];
}

|
//output
document.getElementById('result').innerHTML
L = ('T = ' + Barge.T + ', FB = ' +
Barge.FB);
</script>
```

Intro to JS – Ship as prototype

- Object-oriented
- Ship is a constructor (similar to class)
- Barge is an object of this constructor (similar to instance)
- It can have methods (lwt)

```
<script>
//Ship as object
function Ship (name){
  this.name = name;
  this.L = 100;
  this.B = 40;
  this.D = 20;
  this.lwt = 0.3*this.L*this.B*this.D
}

Barge = new Ship('Barge 1');
ro = 1.025;
W = 5000;
[Barge.T,Barge.FB] = calc_draft_fb(Barge,W,ro)

function calc_draft_fb(ship,w,ro){
  t = (ship.lwt+w)/(ship.L*ship.B*ro);
  fb = ship.D-t;
  return [t,fb];
}

//output
document.getElementById('result').innerHTML = ('T
= ' + Barge.T + ', FB = ' + Barge.FB);
console.log(Barge);
</script>
```


Intro to JS – Barges as instances

- Barge 1 to Barge n
- List of objects

```
<script>
//Ship as object
function Ship (name){
  this.name = name;
  this.L = 100;
  this.B = 40;
  this.D = 20;
  this.lwt = 0.3*this.L*this.B*this.D
}

|
Barge_1 = new Ship('Barge 1');
Barge_2 = new Ship('Barge 2');
Barge_2.L = 120;

//A list of objects
myShips = [Barge_1, Barge_2];

//output
console.log(myShips);
</script>
```

Intro to JS – Large Design Space

- Barge 1 to Barge n
- Vary L (10-200)
- Vary B (10-100)
- Vary D (5-50)
- 769500 Ships
- Create 1 ship for each, with ID
- Show List of Ships

```
function Ship (name){
  this.name = name;
  this.L = 100;
  this.B = 40;
  this.D = 20;
  this.calc_lwt = function (){return 0.3*this.L*this.B*this.D}
}
barge_id = 1;
myShips = [];
//For all L's from 10 to 200, every 1 step
for (l = 10; l < 201 ; l++){
  //For all B's from 5 to 100, every 1 step
  for (b = 10; b < 101 ; b++){
    //For all D's from 5 to 50, every 1 step
    for (d = 5; d < 51 ; d++){
      barge = new Ship(barge_id);
      barge.L = l;
      barge.B = b;
      barge.D = d;
      barge.lwt = barge.calc_lwt();
      myShips.push(barge);
      barge_id++;
    }
  }
}
//output
console.log(myShips);
```

Intro to JS – HTML GUI

- HTML as GUI
- Input fields
- (unnecessary) button

Barge App

Creating a Barge Design Space

Parameters

| | | | |
|--|---------------------------------|---------|----------------------------------|
| L min = | <input type="text" value="10"/> | L max = | <input type="text" value="200"/> |
| B min = | <input type="text" value="5"/> | B max = | <input type="text" value="100"/> |
| D min = | <input type="text" value="5"/> | D max = | <input type="text" value="50"/> |
| <input type="button" value="Create Design Space"/> | | | |

Some Result here

Intro to JS – HTML GUI

```
<body>
<h1>Barge App</h1>
<p>Creating a Barge Design Space</p>
<h2>Parameters</h2>
<br>L min = <input type="text" id="lmin" value=10> L max = <input
type="text" id="lmax" value=200>
<br>B min = <input type="text" id="bmin" value=5> B max = <input
type="text" id="bmax" value=100>
<br>D min = <input type="text" id="dmin" value=5> D max = <input
type="text" id="dmax" value=50>
<br><button onclick="create_space()">Create Design Space</button>

<p id="result">Some Result here</p>

</body>
<script>
```

Intro to JS – Accessing HTML

- Each element in the document has an id
- We can access any property (value, text, event)
- Button calls the function
- Show first 1000 designs

```
function create_space (){
    lmin = parseInt(document.getElementById("lmin").value);
    lmax = parseInt(document.getElementById("lmax").value) + 1;
    bmin = parseInt(document.getElementById("bmin").value);
    bmax = parseInt(document.getElementById("bmax").value) + 1;
    dmin = parseInt(document.getElementById("dmin").value);
    dmax = parseInt(document.getElementById("dmax").value) + 1;

    barge_id = 0;
    myShips = [];
    for (l = lmin; l < lmax ; l++){
        for (b = bmin; b < bmax ; b++){
            for (d = dmin; d < dmax ; d++){
                barge = new Ship(barge_id);
                barge.L = l;
                barge.B = b;
                barge.D = d;
                barge.lwt = barge.calc_lwt();
                myShips.push(barge);
                barge_id++;
            }
        }
    }
    //output
    designs = '';
    document.getElementById("result").innerHTML = "Total designs: " + myShips.length;
    for (i = 0; i < 1000; i++){designs = designs + JSON.stringify(myShips[i])};
    document.getElementById("designs").innerHTML = designs;
}
</script>
```

Intro to JS – Accessing HTML

Barge App

Creating a Barge Design Space

Parameters

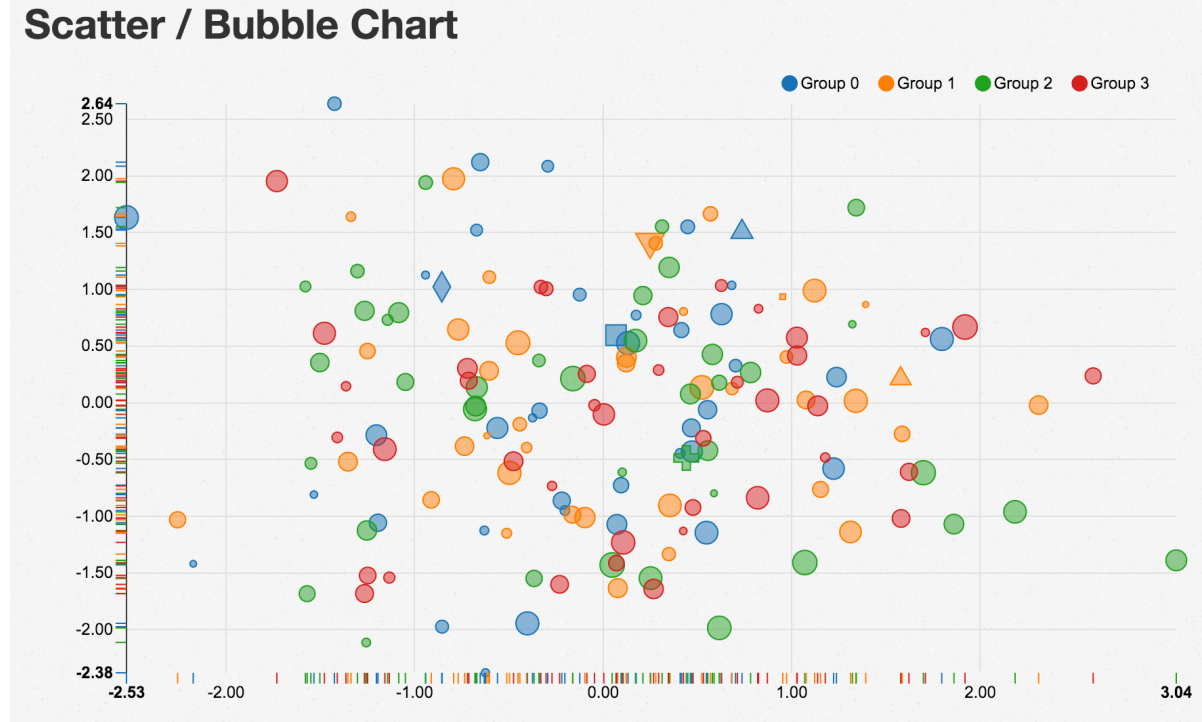
| | | | |
|-------------------------------------|----|---------|-----|
| L min = | 10 | L max = | 200 |
| B min = | 10 | B max = | 100 |
| D min = | 5 | D max = | 50 |
| Create Design Space | | | |

Total designs: 799526

```
{"name":0,"L":10,"B":10,"D":5,"lwt":150}{ "name":1,"L":10,"B":10,"D":6,"lwt":180}{ "name":2,"L":10,"B":10,"D":7,"lwt":210}{ "name":3,"L":10,"B":10,"D":8,"lwt":240}
{"name":4,"L":10,"B":10,"D":9,"lwt":270}{ "name":5,"L":10,"B":10,"D":10,"lwt":300}{ "name":6,"L":10,"B":10,"D":11,"lwt":330}{ "name":7,"L":10,"B":10,"D":12,"lwt":360}
{"name":8,"L":10,"B":10,"D":13,"lwt":390}{ "name":9,"L":10,"B":10,"D":14,"lwt":420}{ "name":10,"L":10,"B":10,"D":15,"lwt":450}{ "name":11,"L":10,"B":10,"D":16,"lwt":480}
{"name":12,"L":10,"B":10,"D":17,"lwt":510}{ "name":13,"L":10,"B":10,"D":18,"lwt":540}{ "name":14,"L":10,"B":10,"D":19,"lwt":570}{ "name":15,"L":10,"B":10,"D":20,"lwt":600}
{"name":16,"L":10,"B":10,"D":21,"lwt":630}{ "name":17,"L":10,"B":10,"D":22,"lwt":660}{ "name":18,"L":10,"B":10,"D":23,"lwt":690}{ "name":19,"L":10,"B":10,"D":24,"lwt":720}
{"name":20,"L":10,"B":10,"D":25,"lwt":750}{ "name":21,"L":10,"B":10,"D":26,"lwt":780}{ "name":22,"L":10,"B":10,"D":27,"lwt":810}{ "name":23,"L":10,"B":10,"D":28,"lwt":840}
{"name":24,"L":10,"B":10,"D":29,"lwt":870}{ "name":25,"L":10,"B":10,"D":30,"lwt":900}{ "name":26,"L":10,"B":10,"D":31,"lwt":930}{ "name":27,"L":10,"B":10,"D":32,"lwt":960}
{"name":28,"L":10,"B":10,"D":33,"lwt":990}{ "name":29,"L":10,"B":10,"D":34,"lwt":1020}{ "name":30,"L":10,"B":10,"D":35,"lwt":1050}{ "name":31,"L":10,"B":10,"D":36,"lwt":1080}
{"name":32,"L":10,"B":10,"D":37,"lwt":1110}{ "name":33,"L":10,"B":10,"D":38,"lwt":1140}{ "name":34,"L":10,"B":10,"D":39,"lwt":1170}{ "name":35,"L":10,"B":10,"D":40,"lwt":1200}
{"name":36,"L":10,"B":10,"D":41,"lwt":1230}{ "name":37,"L":10,"B":10,"D":42,"lwt":1260}{ "name":38,"L":10,"B":10,"D":43,"lwt":1290}{ "name":39,"L":10,"B":10,"D":44,"lwt":1320}
{"name":40,"L":10,"B":10,"D":45,"lwt":1350}{ "name":41,"L":10,"B":10,"D":46,"lwt":1380}{ "name":42,"L":10,"B":10,"D":47,"lwt":1410}{ "name":43,"L":10,"B":10,"D":48,"lwt":1440}
{"name":44,"L":10,"B":10,"D":49,"lwt":1470}{ "name":45,"L":10,"B":10,"D":50,"lwt":1500}{ "name":46,"L":10,"B":11,"D":5,"lwt":165}{ "name":47,"L":10,"B":11,"D":6,"lwt":198}
{"name":48,"L":10,"B":11,"D":7,"lwt":231}{ "name":49,"L":10,"B":11,"D":8,"lwt":264}{ "name":50,"L":10,"B":11,"D":9,"lwt":297}{ "name":51,"L":10,"B":11,"D":10,"lwt":330}
{"name":52,"L":10,"B":11,"D":11,"lwt":363}{ "name":53,"L":10,"B":11,"D":12,"lwt":396}{ "name":54,"L":10,"B":11,"D":13,"lwt":429}{ "name":55,"L":10,"B":11,"D":14,"lwt":462}
{"name":56,"L":10,"B":11,"D":15,"lwt":495}{ "name":57,"L":10,"B":11,"D":16,"lwt":528}{ "name":58,"L":10,"B":11,"D":17,"lwt":561}{ "name":59,"L":10,"B":11,"D":18,"lwt":594}
{"name":60,"L":10,"B":11,"D":19,"lwt":627}{ "name":61,"L":10,"B":11,"D":20,"lwt":660}{ "name":62,"L":10,"B":11,"D":21,"lwt":693}{ "name":63,"L":10,"B":11,"D":22,"lwt":726}
{"name":64,"L":10,"B":11,"D":23,"lwt":759}{ "name":65,"L":10,"B":11,"D":24,"lwt":792}{ "name":66,"L":10,"B":11,"D":25,"lwt":825}{ "name":67,"L":10,"B":11,"D":26,"lwt":858}
{"name":68,"L":10,"B":11,"D":27,"lwt":891}{ "name":69,"L":10,"B":11,"D":28,"lwt":924}{ "name":70,"L":10,"B":11,"D":29,"lwt":957}{ "name":71,"L":10,"B":11,"D":30,"lwt":990}
{"name":72,"L":10,"B":11,"D":31,"lwt":1023}{ "name":73,"L":10,"B":11,"D":32,"lwt":1056}{ "name":74,"L":10,"B":11,"D":33,"lwt":1089}{ "name":75,"L":10,"B":11,"D":34,"lwt":1122}
{"name":76,"L":10,"B":11,"D":35,"lwt":1155}{ "name":77,"L":10,"B":11,"D":36,"lwt":1188}{ "name":78,"L":10,"B":11,"D":37,"lwt":1221}{ "name":79,"L":10,"B":11,"D":38,"lwt":1254}
{"name":80,"L":10,"B":11,"D":39,"lwt":1287}{ "name":81,"L":10,"B":11,"D":40,"lwt":1320}{ "name":82,"L":10,"B":11,"D":41,"lwt":1353}{ "name":83,"L":10,"B":11,"D":42,"lwt":1386}
```

Intro to JS – Plotting

- Many Libraries
- Nvd3 as example
- Easy to use and based on D3.js
- <http://nvd3.org/examples/scatter.html>



Intro to JS – Plotting

- Download library and dependencies
- Include in your `<head></head>`
- Include example in your code
- Adapt example to what you want to plot

```
<head>
<link href="nv.d3.css" rel="stylesheet">
<script src="d3.v3.min.js"></script>
<script src="nv.d3.js"></script>
</head>
```

```
//plot
nv.addGraph(function() {
  var chart = nv.models.scatterChart()
    .showDistX(true) //showDist, when true,
    //will display those little distribution lines on the axis.
    .showDistY(true)
    .transitionDuration(350)
    .color(d3.scale.category10().range());

  //Configure how the tooltip looks.
  chart.tooltipContent(function(key) {
    return '<h3>' + key + '</h3>';
  });

  //Axis settings
  chart.xAxis.tickFormat(d3.format('.02f'));
  chart.yAxis.tickFormat(d3.format('.02f'));

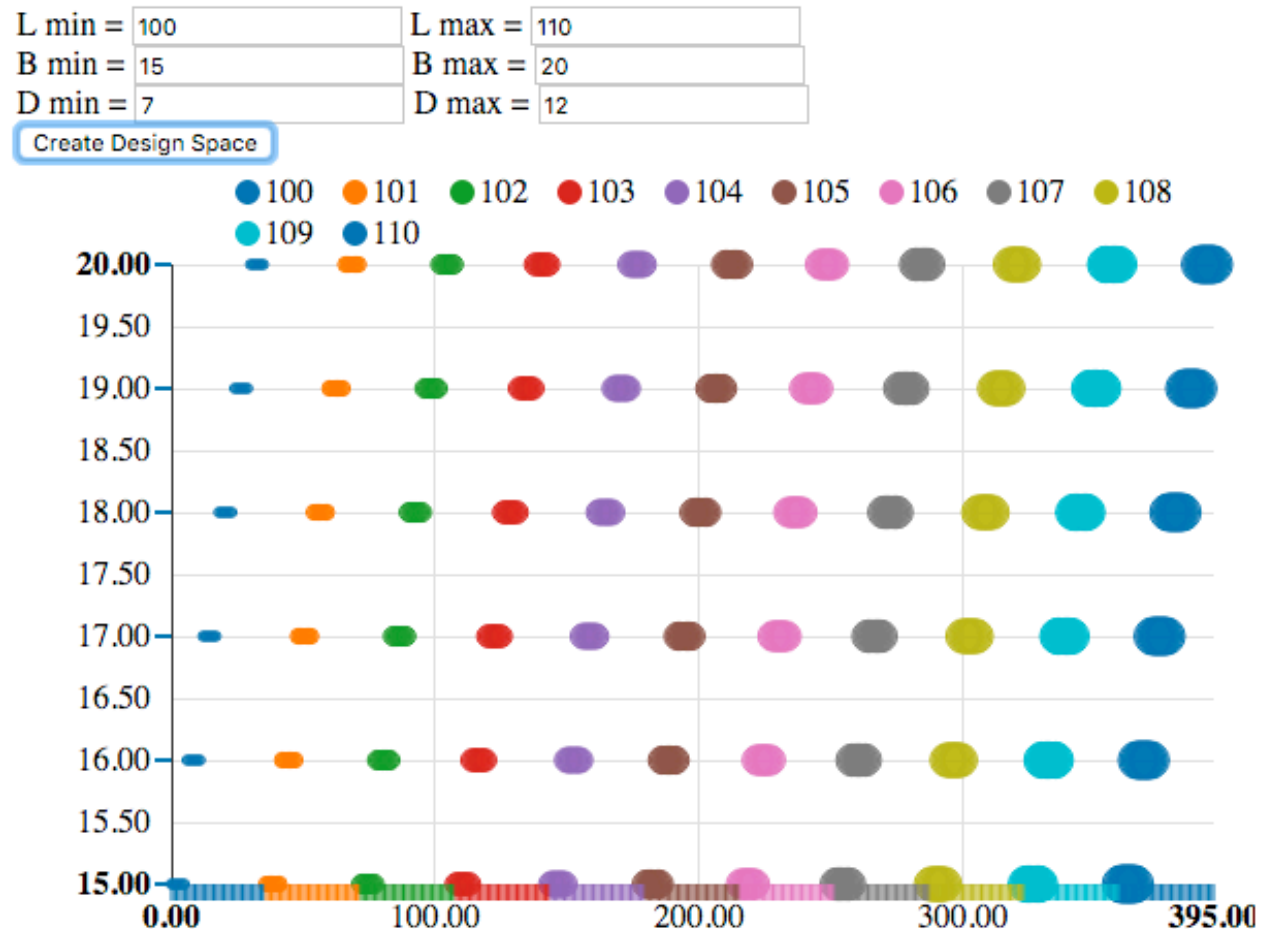
  d3.select('#chart svg')
    .datum(myData)
    .call(chart);

  nv.utils.windowResize(chart.update);

  return chart;
});
```

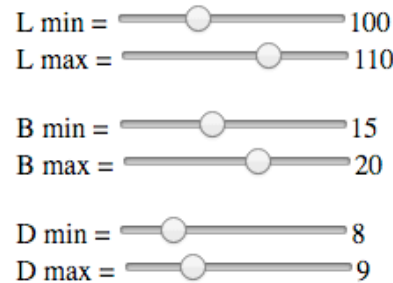

Intro to JS – Plotting

- Multi-series
- Design set “n” (Length)
- Data to plot created inside the same “for” loop

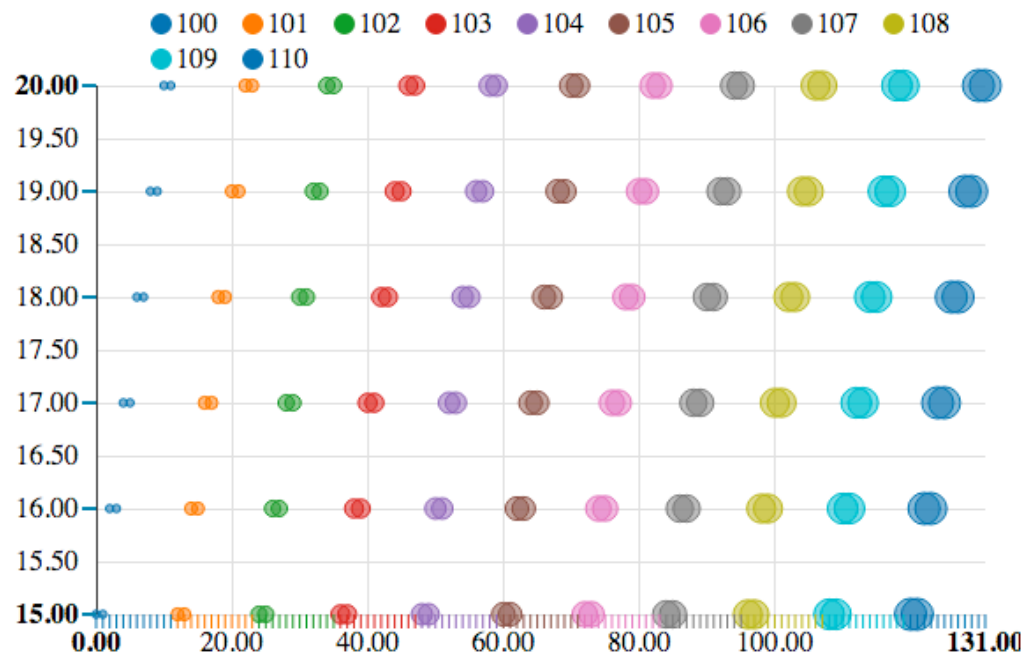


Intro to JS – Plotting

- Range slider
- Removing the button
- Calling function “oninput”
- Calling when loading the page
- This interactivity is the main bonus of javascript – no other GUI provides so much freedom



Total designs: 132



Intro to JS – SVG

- SVG is one option to draw 2D in a browser
- `<svg></svg>`
- From basic shape to any complex vector shape
- Standard to many other 2D software

```
<svg> <circle cx="50" cy="50" r="40" stroke="green" stroke-width="4" fill="yellow" /></svg>
```

Building Block App



Intro to JS – SVG

- SVG, as any other element in the page, is highly interactive
- Click to execute a function, for example resize
- This principle is the basis for more complex libraries, such as D3

```
<svg> <circle id="circle_2" onclick="size()"
cx="50" cy="50" r="40" stroke="green" stroke-
width="4" fill="yellow" /></svg>
</body>
<script>
function size(){
console.log(document.getElementById("circle_2").se
tAttribute('r',String(Math.random()*100)));
}
</script>
```

Building Block App



Case Study – Holtrop Method

<http://shiplab.hials.org/app/holtrop/>

Holtrop calculation

Hull parameters:

LWL: 252
 Breadth: 42
 Draught Foreward: 14
 Draught Aftward: 14
 LCB (%): 3.1
 CB: 0.85
 Midship Coefficient: 1
 CWL: 0.91
 Service Speed: 15

Bulbous?: 1 Yes = 1
 Transom?: 1 No = 0

Afterbody form: 3
 Pram with Gondola = 1
 V-Shaped Sections = 2
 Normal Section Shape = 3
 U-Shaped Sections with Hogues Stern = 4

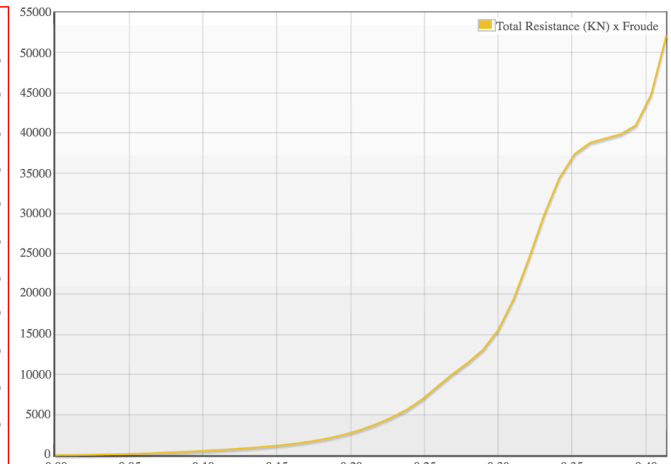
Appendage parameters:

Rudder behind Skeg: 1.5 Area: 0
 Rudder behind Stern: 1.3 Area: 0
 Twin-Screw Balance Rudders (2.8) Area: 0
 Shaft Brackets (3.0) Area: 0
 Skeg: 1.5 Area: 0
 Strut Bossings (3.0) Area: 0
 Hull Bossings (2.0) Area: 0
 Shafts: 2 Area: 0
 Stabilizer Fins (2.8) Area: 0
 Dome (2.7) Area: 0
 Bilge Keel (1.4) Area: 0

$$\text{Appendage Resistance Factor} : (1 + k_2)_{eq} = \frac{\sum (1 + k_2) S_{App}}{\sum S_{App}}$$

Holtrop is a method based on statistical regression of model tests and results from ship trials. The method is used to estimate the resistance of displacement ships. The database covers a wide range of ships. For extreme shapes, the accuracy of the estimates is not good. The method may be used to assess qualitatively the resistance of a ship design. [More](#)

Developed by: Jefferson Flor, Thiago Gabriel Monteiro, Henrique M. Gaspar ([ShipLab web app](#)).



| Resistance | Value (kN) |
|-------------|------------|
| Total | 1271.4 |
| Frictional | 676.7 |
| Form | 250.2 |
| Appendage | 0.0 |
| Wave | 100.1 |
| Bulbous | 0.1 |
| Transom | 116.9 |
| Correlation | 127.3 |

Case Study – Parametric Design

<http://uscience.org/files/parametric.html>

The problem is to design an AHTS for the support of offshore operations. The purpose of the design is narrowed to supply, anchor handling and towing missions. More industrial related purposes of OSV design are discussed by Ulstein and Brett [5,6]. Each mission is considered a set of operational profiles, with minimal requirements related to the task activity, such as: supply capacity (e.g. cargo volume $\geq 5000\text{m}^3$ and cargo area $\geq 500\text{m}^2$), field operations requirements (e.g. bollard pull (e.g. $\geq 200\text{ton}$), illustrated in Figure 2.

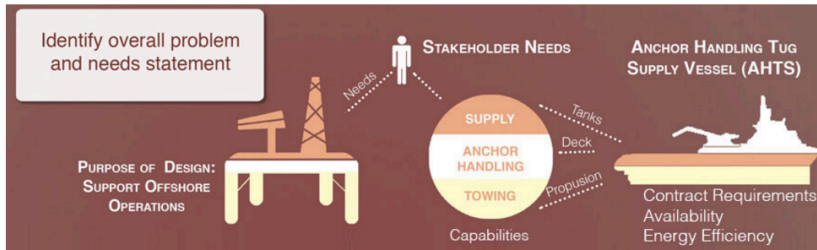


Figure 2 - AHTS link between stakeholder's expectations and mission performance

An OSV mission is decomposed in three operational profiles: anchor handling, supplying and towing. E operational states' requirements are connected to three vessel capabilities: cargo deck area [m^2], bollard pull [MT] and cargo volume [m^3].

Table 1 - Mission, requirements and capabilities

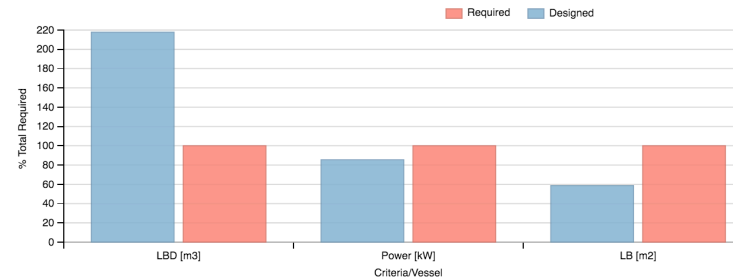
| Capability | A. Handling | Supplying | Towing | Total |
|----------------------------------|-------------|-----------|--------|-------|
| Cargo deck area [m^2] | 500 | 500 | 500 | 1500 |
| Bollard pull [MT] | 50 | 0 | 200 | 250 |
| Cargo volume [m^3] | 500 | 3000 | 500 | 4000 |

The value of the table connects mission requirements with vessel capabilities. By changing any of the values, the total required capability changes, modifying the criteria and the requirement dependency wheel below.

Table 3 - Vessel parameters

| Parameters | Value | Range |
|----------------------|-------|-------|
| Length [m] | 100 | |
| Breadth/Length | 0.22 | |
| Breadth [m] | 22 | |
| Depth/Breadth | 0.45 | |
| Depth [m] | 9.9 | |
| Draft/Depth | 0.7 | |
| Draft/Design [m] | 6.9 | |
| C_B | 0.75 | |
| Powering [kW] | 20689 | |
| Price / GT [kNOK/GT] | 50 | |

By sliding the bars and modifying the vessel parameters we are able to change its capabilities, and consequently evaluate if the criteria is met.



Case Study – 6 dof Ship + masses

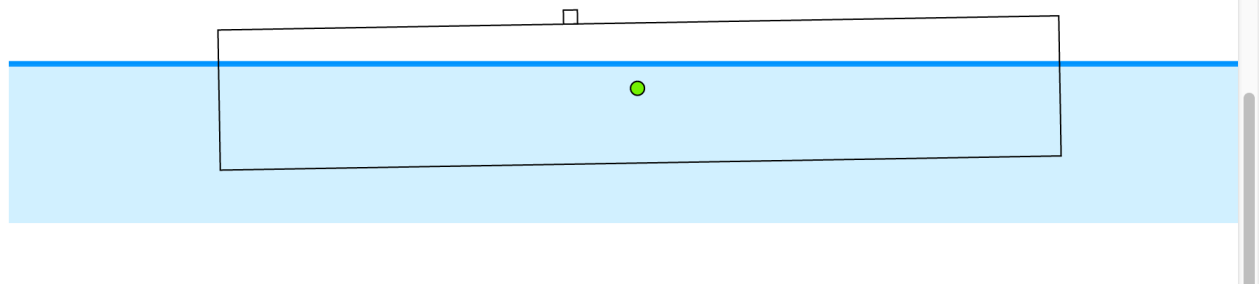
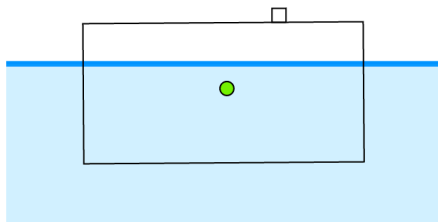
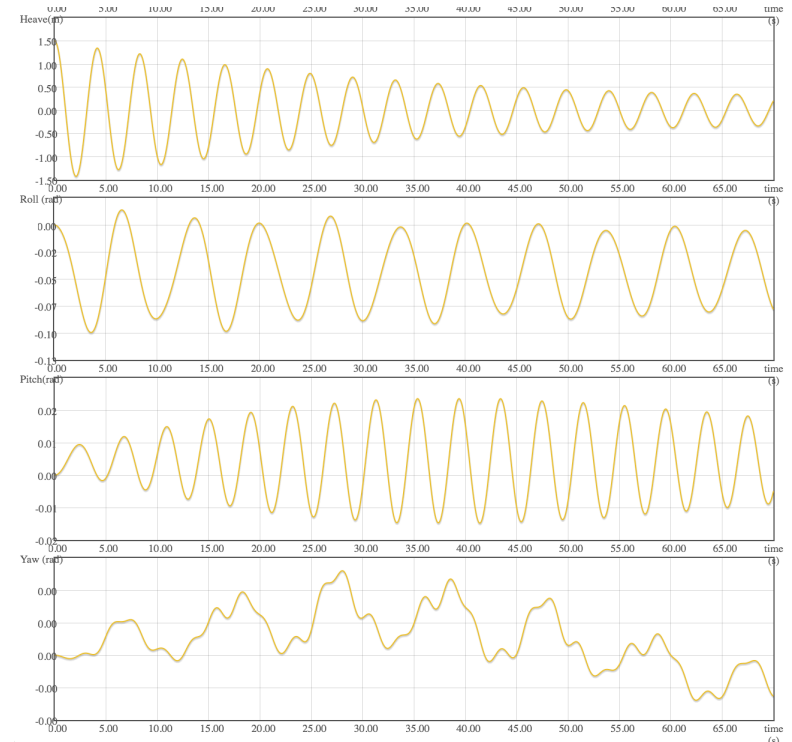
<http://www.shiplab.hials.org/app/6dof/>

Parameters:

| | | |
|---|---------|-----------------------|
| Length Waterline | 60 | <input type="range"/> |
| Breadth | 20 | <input type="range"/> |
| Depth | 10 | <input type="range"/> |
| Draft | 6 | <input type="range"/> |
| Block Coefficient : | 0.67 | <input type="range"/> |
| KG | 5 | <input type="range"/> |
| Time Span | 70 | <input type="range"/> |
| Linear Resist. Coef. [m/s] | 0.2 | <input type="range"/> |
| Roll Lin. Damp. Coef. [kgm ² /s] | 3000000 | <input type="range"/> |
| Initial Heave. [m] | 1.5 | <input type="range"/> |
| Initial Roll. [rad] | 0 | <input type="range"/> |
| Initial Pitch. [rad] | 0 | <input type="range"/> |

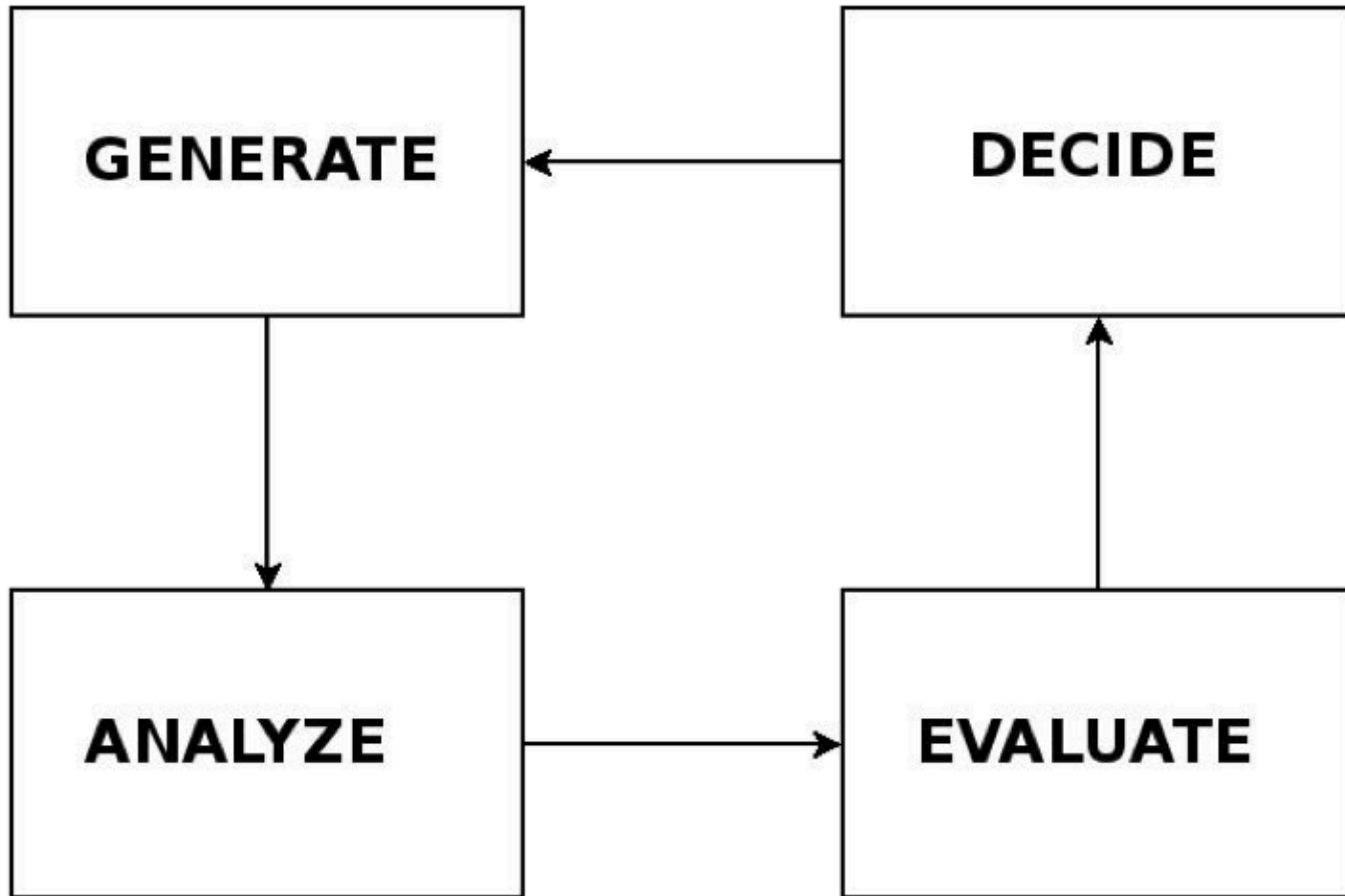
Simulation time: 26.73s

Frame rate: 17.53 FPS



III – DESIGN AND ENGINEERING

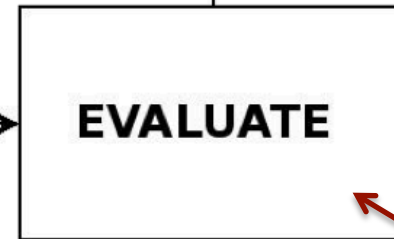
BASIC DESIGN PROCESS



BASIC DESIGN PROCESS - JS

Library of X:

- Ship
- Blocks
- Systems
- Yards
- Services
- Methods



Selection of α :

- Choice
- Perception
- Handling
- Re-analysis

Calculation of Y:

- Cost
- Performance
- Comparison
- Arrangement
- Optimisation

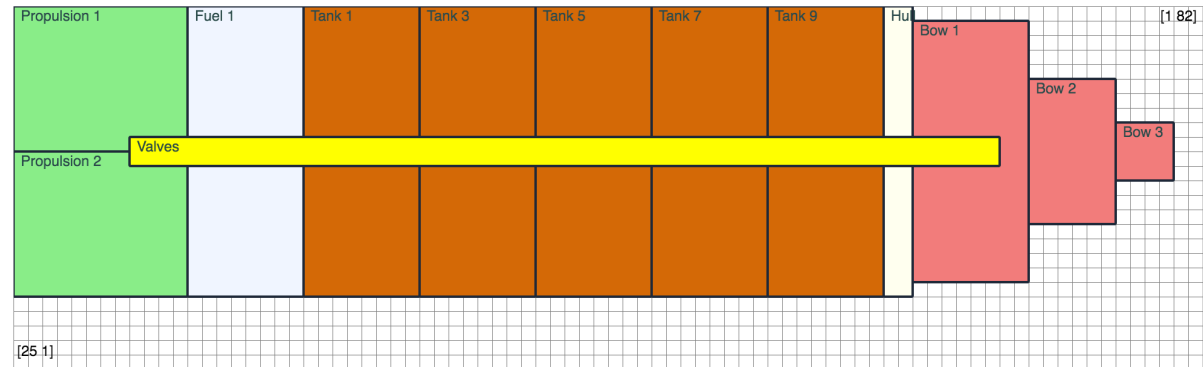
Visualization of Z:

- Rank (Best among peers)
- Tradeoffs
- Charts
- Drawings
- Solutions

Creating Libraries - JS

- Library of X:

- Ship
- Blocks
- Systems
- Yards
- Services
- Methods()



- Every X is an Object {}

- Properties - size:100; name: 'UCL'; subsystems = [{sub1},{sub2}]
- Methods - run(); sail(); dance();

- We can access and interact with any part of this object
- Interaction among multi-libraries
- Think about your Xs – libraries of what?

Analysing Objects - JS

- Calculation of Y:
 - Cost
 - Performance
 - Comparison
 - Arrangement
 - Optimisation
- Every Y is handled by a function / Method ()
 - `Cost(ship) = 1000`
 - `Cost(ship.propulsion) = 100`
 - `ShipA.operability(NorthSea) = 99%`
- It can interact with your library (object), altering it or creating new data
- Think about your Ys – analyses of what?

Space Information:

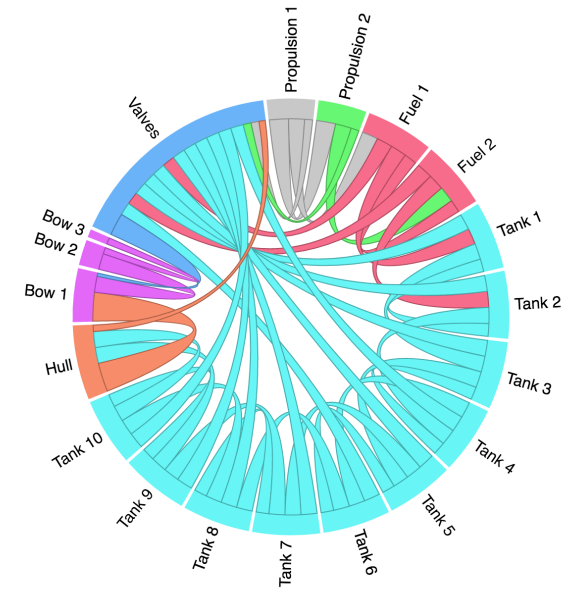
Columns: 82
Rows: 25
Total Area: 2050

Matches:

Propulsion 1, Propulsion 2 - 16 grid spaces
Propulsion 1, Cargo - 8 grid spaces
Propulsion 2, Propulsion 1 - 16 grid spaces
Propulsion 2, Cargo - 8 grid spaces
Cargo, Propulsion 1 - 8 grid spaces
Cargo, Propulsion 2 - 8 grid spaces
Cargo, Bow - 16 grid spaces
Bow, Cargo - 16 grid spaces
Crane, Cargo - 32 grid spaces

Evaluation of Data- JS

- Visualization of Z:
 - Rank (Best among peers)
 - Tradeoffs
 - Charts
 - Drawings
 - Solutions
- From data to information, from information to knowledge
- Every Z is knowledge extracted from your data
 - Scatter plots
 - Pareto Frontier
 - Design that fits better a certain scenario
- It relies heavily on the visual information - visual displays provide the highest bandwidth channel from the computer to the human
- Usually the most time consuming part – dealing with GUI, with personal options, preferences, resistance
- Think about your Zs – evaluation of what?



Decision Making - JS

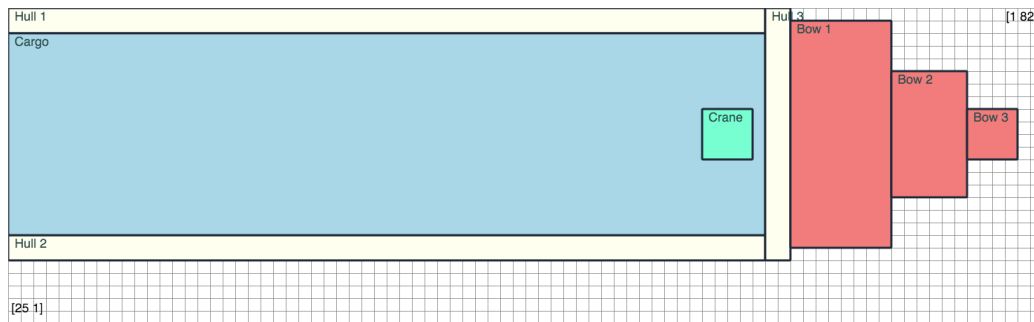
- Selection of a:
 - Choice
 - Perception
 - Handling
 - Re-analysis
- Bounded rationality
- Every α is decision based on a rational logic
- Guesses about uncertain future consequences (of current actions)
- Guesses about uncertain future preferences (for those consequences)
- Satisficing (rather than max/min) process – good enough
- Computer as tool.
- JS method to facilitate a new loop (re-generation, re-analyses, re-evaluation)
- Think about your α_s – decision based on what?



Case Study – Layout Tool

<http://uscience.org/files/grid/>

Database: ☐ simple_blocks.csv ☒ maindeck.csv ☐ twindeck.csv



Space Information:

Columns: 82
Rows: 25
Total Area: 2050

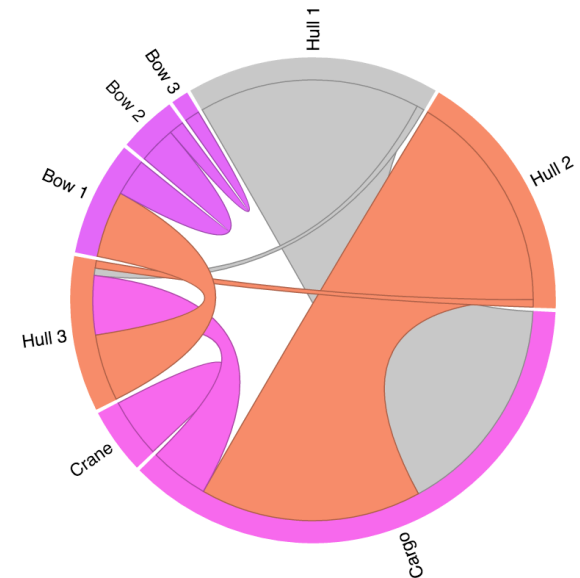
Matches:

Hull 1, Cargo - 60 grid spaces
Hull 1, Hull 3 - 2 grid spaces
Hull 2, Cargo - 60 grid spaces
Hull 2, Hull 3 - 2 grid spaces
Cargo, Hull 1 - 60 grid spaces
Cargo, Hull 2 - 60 grid spaces
Cargo, Hull 3 - 16 grid spaces
Crane, Cargo - 16 grid spaces
Hull 3, Hull 1 - 2 grid spaces
Hull 3, Hull 2 - 2 grid spaces
Hull 3, Cargo - 16 grid spaces
Hull 3, Bow 1 - 18 grid spaces
Bow 1, Hull 3 - 18 grid spaces
Bow 1, Bow 2 - 10 grid spaces
Bow 2, Bow 1 - 10 grid spaces
Bow 2, Bow 3 - 4 grid spaces
Bow 3, Bow 2 - 4 grid spaces

Number of Blocks in the Database: 8

Block Information (click any block):

ID:
Width: 4
Height: 4
Block Area: 16
Space Used:
[9, 77], [9, 78], [9, 79], [9, 80], [10, 77], [10, 78], [10, 79], [10, 80], [11, 77], [11, 78], [11, 79], [11, 80], [12, 77], [12, 78], [12, 79], [12, 80],



Databases: [simple_block.csv](#) , [maindeck.csv](#) , [twindeck.csv](#)

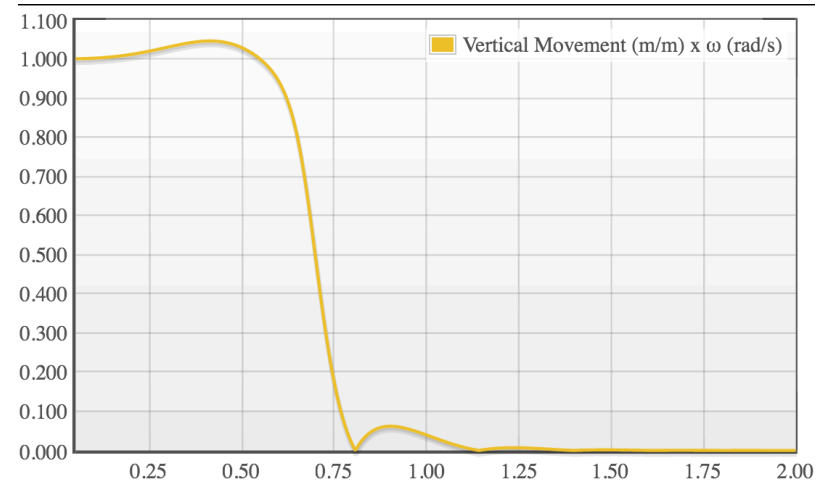
Introduction to JavaScript Applied to Design and Engineering

an informal talk at UCL

April 21st 2016, London UK

Parameters for user input

| | | | |
|----------------------------------|----------------------------------|---|-------------------------------------|
| Area of tank 1 [m ²] | <input type="text" value="100"/> | Pipe diameter [m] | <input type="text" value="0.5"/> |
| Area of tank 2 [m ²] | <input type="text" value="100"/> | Pipe roughness [m] | <input type="text" value="0.0002"/> |
| Initial Height of tank 1 [m] | <input type="text" value="6"/> | Liquid density [kg/m ³] | <input type="text" value="1000"/> |
| Initial Height of tank 2 [m] | <input type="text" value="4"/> | Liquid viscosity | <input type="text" value="0.0001"/> |
| Final Height of tank 1 [m] | <input type="text" value="8"/> | Acceleration of Gravity [m/s ²] | <input type="text" value="9.81"/> |
| Final Height of tank 2 [m] | <input type="text" value="2"/> | Simulation time [s] | <input type="text" value="100"/> |
| Pipe length [m] | <input type="text" value="20"/> | Pump Capacity [m ³ /s] | <input type="text" value="1"/> |



Assoc. Prof. Henrique M. Gaspar, PhD

Faculty of Maritime Technology and Operations - NTNU

henrique.gaspar@ntnu.no