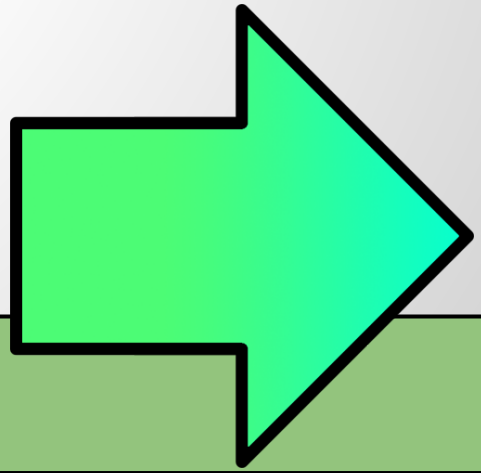
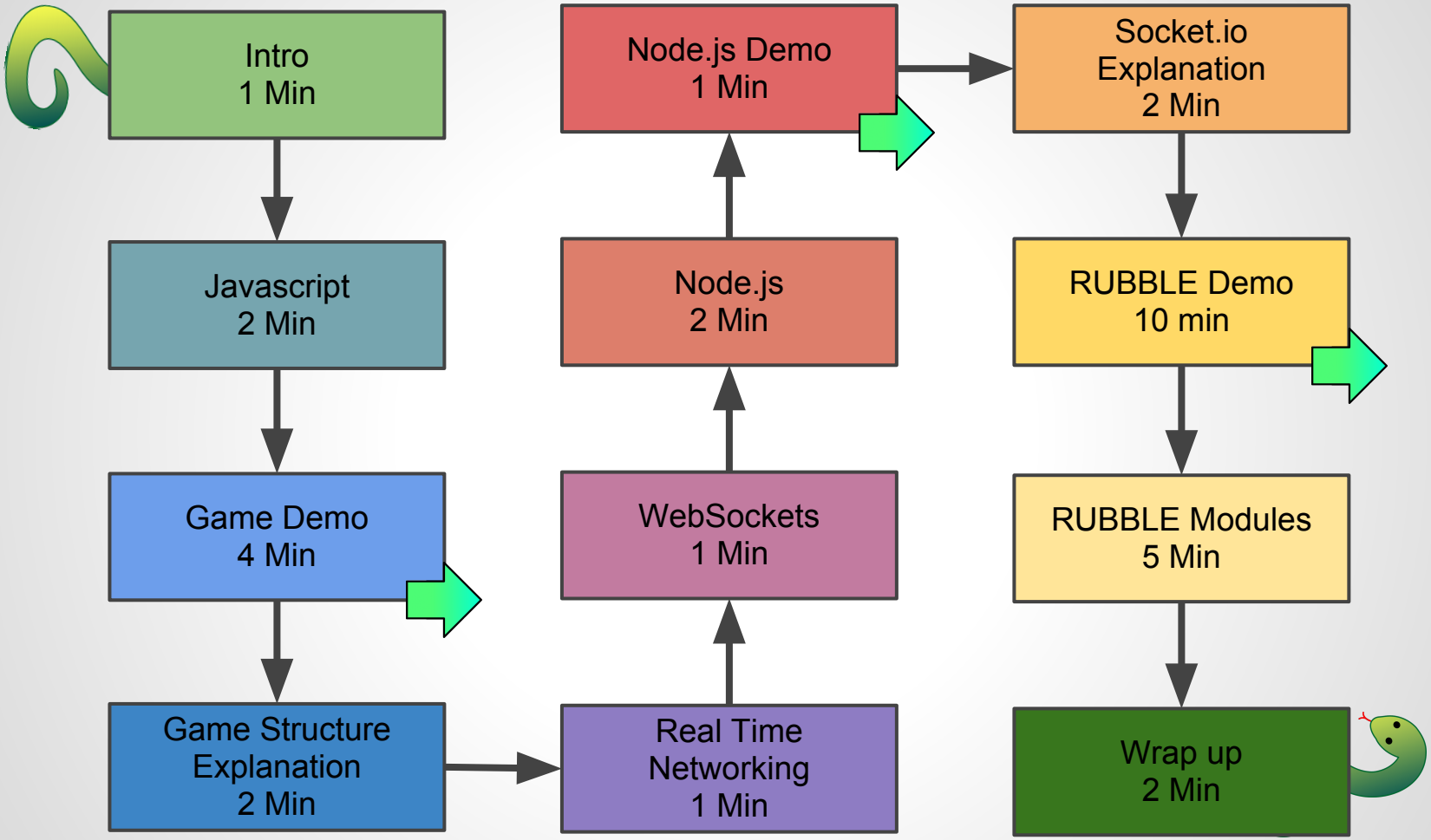
RUBBLE

By Nick and Sam



Our Presentation



About our project

- The Suggestion
    - Make a real time application
    - Use cutting edge technology
- The Project
    - A video game
    - Social interactions
- Why?
    - Extremely challenging
    - A Childhood Dream

Platform / technology

- Why the Web
  - Accessibility
  - Becoming more dominant

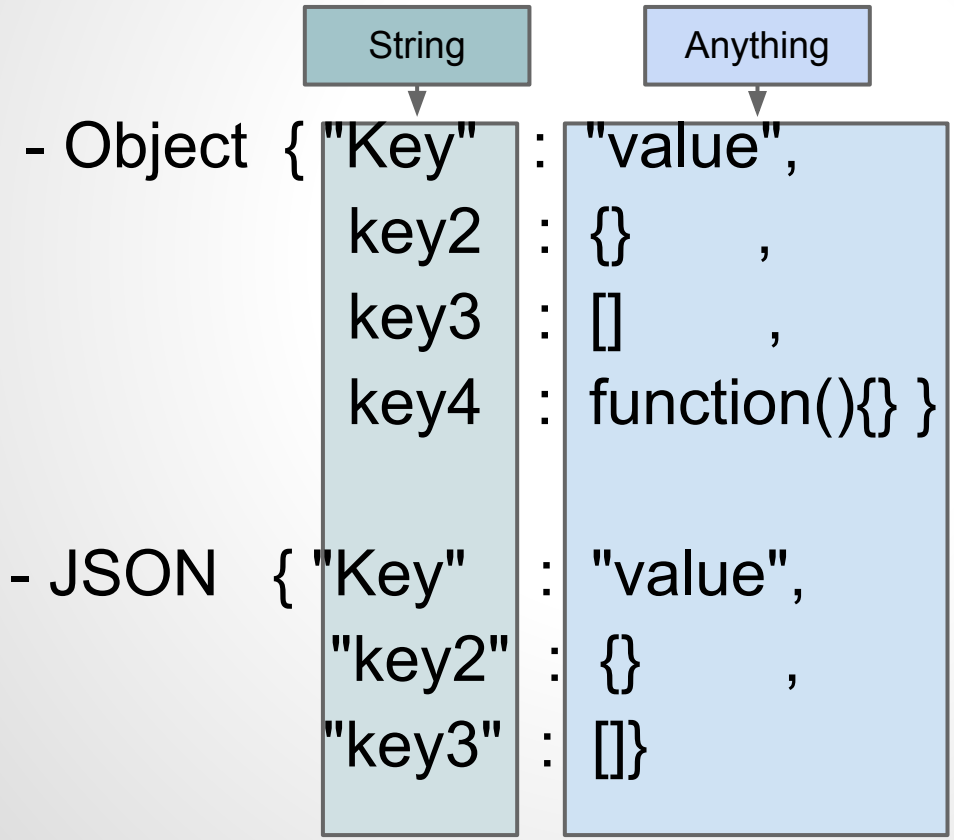
- ecmascript the language of the web (Javascript)
  - Because we had to
  - It's not bad when you get used to it
- node.js
  - Uses javascript
  - WebSockets
  - Amazing Docs and Package Manager

Getting used to Javascript

- Origins
    - Brendan Eich
    - 10 Days
- Popularity
    - Becoming popular
- Differences with classical OO languages.
    - Super versatile
    - Lacks structure

Getting used to Javascript

Part 1: Javascript Objects



Inline objects

```
var cat = {
    paws : 4,
    tail : 1,
    sound : "Meow!"
};
```

Modifying properties

```
//Accident occurs
cat.paws = 3;

//Assignements
cat.sound = "Woof!";
cat.sound = 42;
cat.sound = [1,2,3,4];
cat.sound = cat;
```

Modyfing objects

```
//Adding a property
cat.name = "Kitty";

//deleting a property
delete cat.tail;
```

Getting used to Javascript

Part 2: Funky Functions

-Functions are:

- Objects
- Classes
- Constructors
- Modules

```
//function as a constructor and class
function Cat(catName,catColor) {
    this.name = catName;
    this.color = catColor;
    this.paws = 4;
    this.sound = "Meow!";

    //function as a method
    this.talk = function () {
        console.log(this.sound);
    }
}

var cat = new Cat('Kitty','Blood Red');
```

Getting used to Javascript

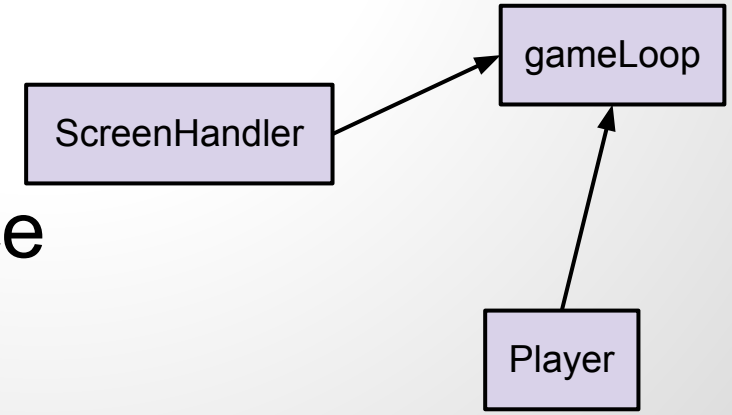
Part 3: Organization

-All variables are global scope

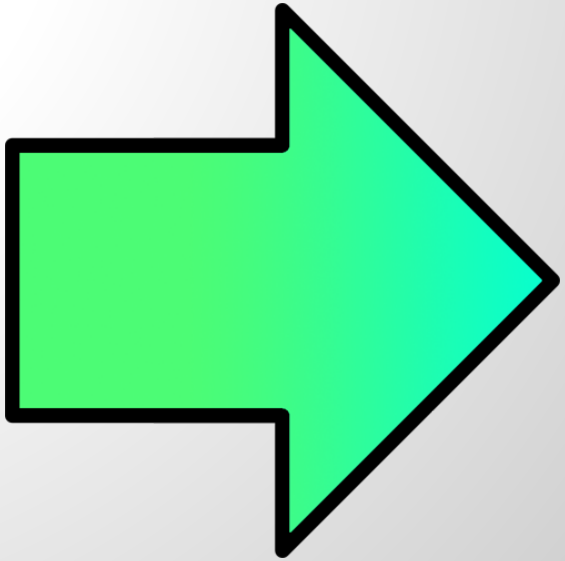
-Libraries to fix this (modules)

- RequireJs
- CommonJs

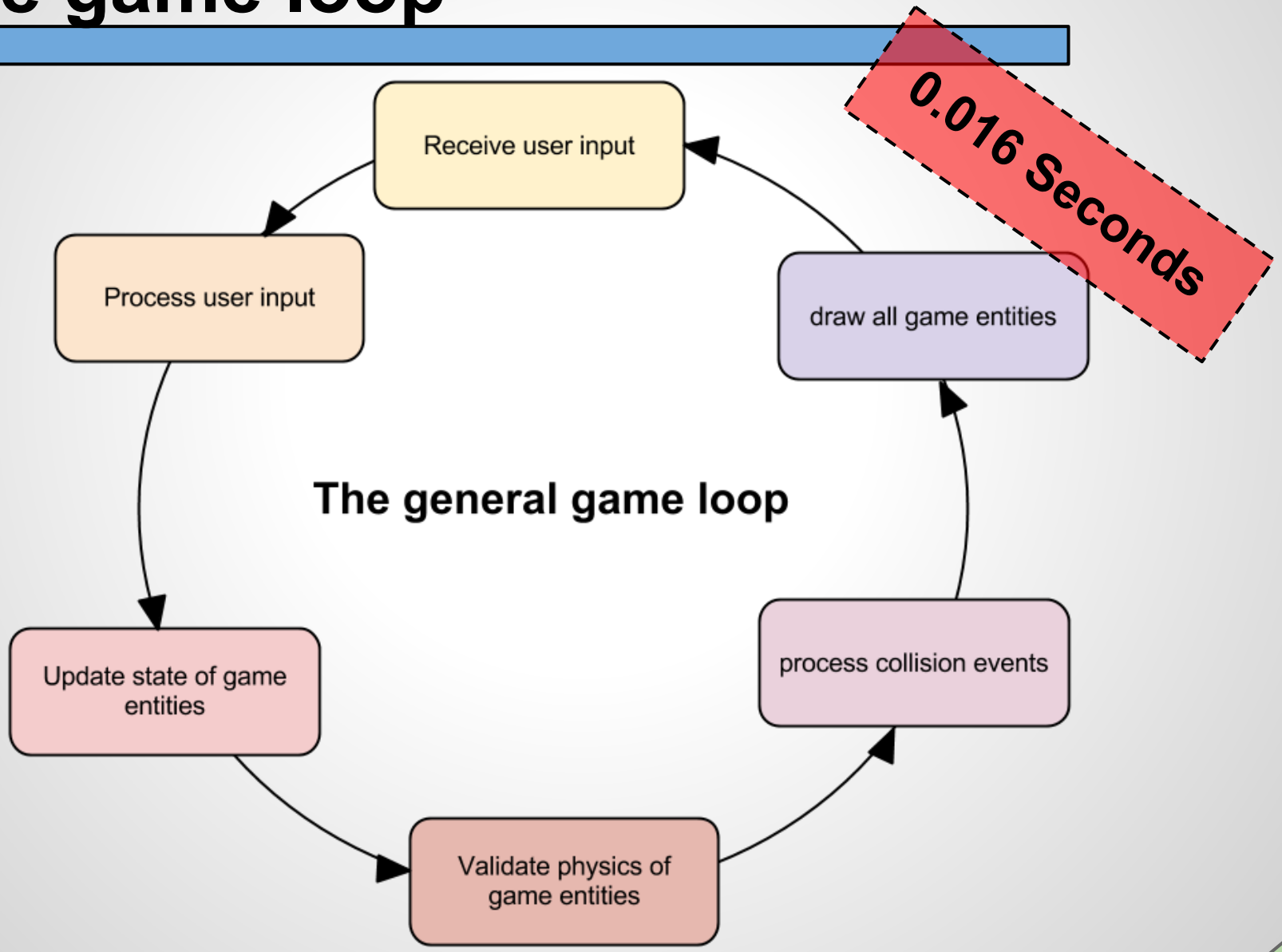
-Can't circular reference



Demo - Early build



The game loop

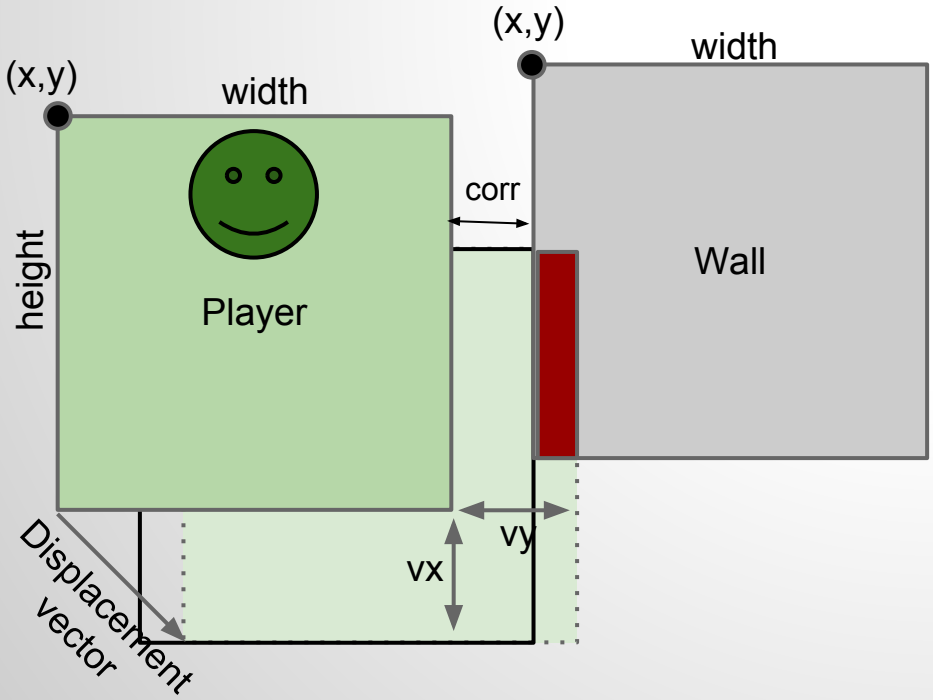


Collisions

Collision detection and resolution:

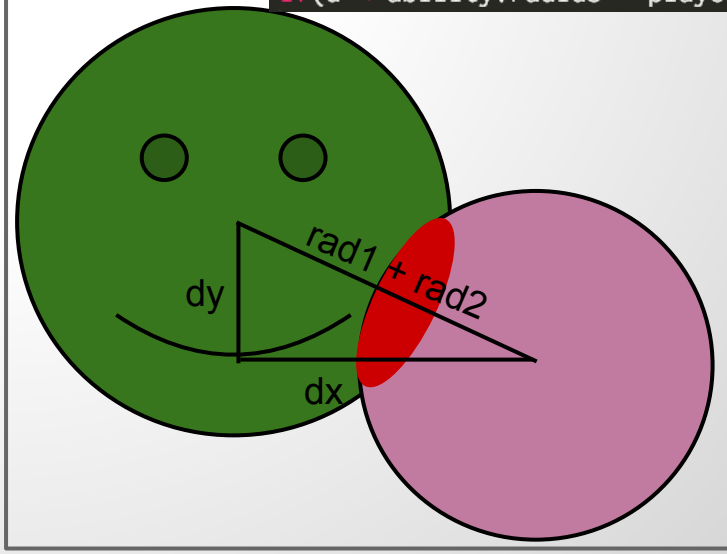
-Squares

```
//check if future player intersects with wall
if(((player.x + player.vx + player.radius > wall.x) && (player.x + player.vx - player.radius < wall.x + wall.width )) &&
   ((player.y + player.vy + player.radius > wall.y) && (player.y + player.vy - player.radius < wall.y + wall.height))){
```



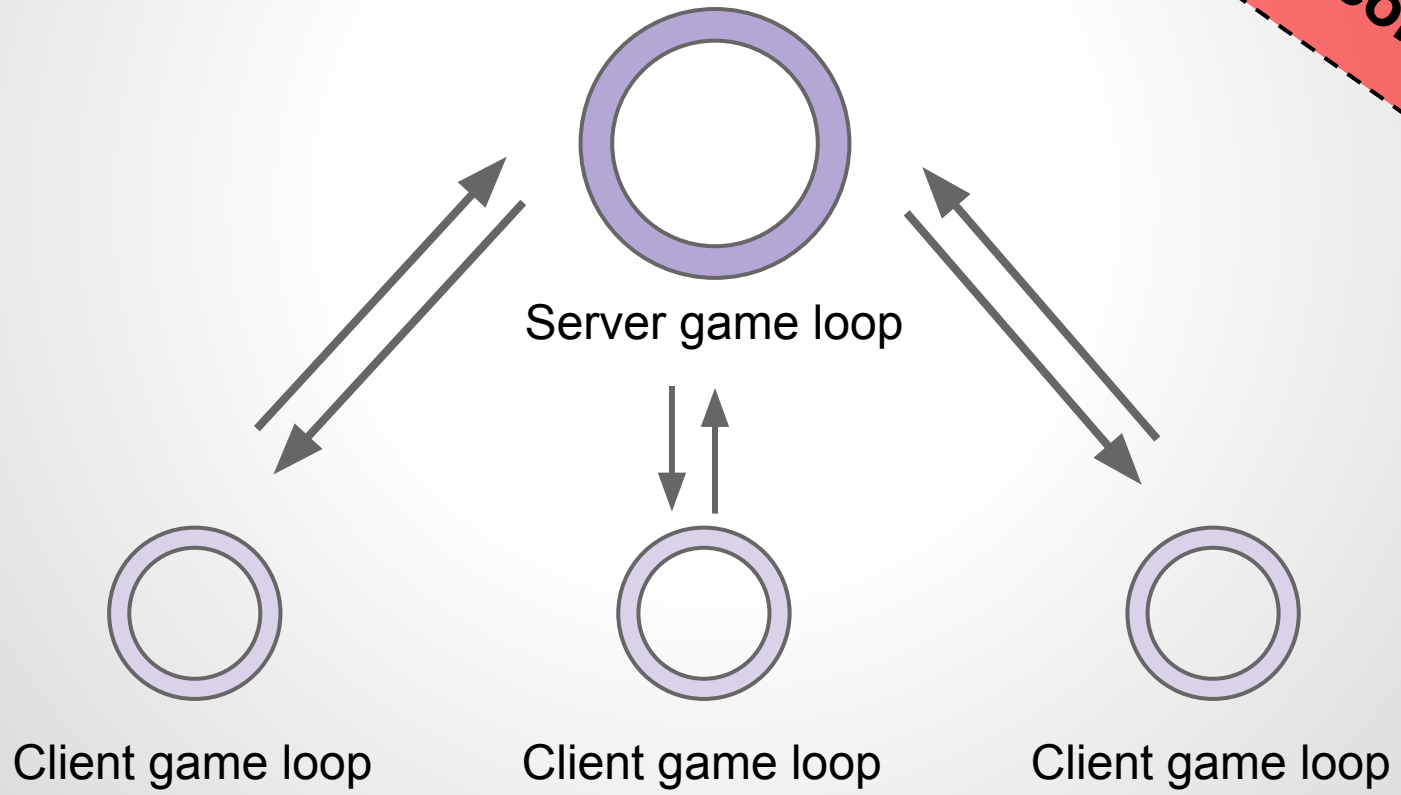
-Circles

```
var dx = Math.abs(player.x - ability.x);
var dy = Math.abs(player.y - ability.y);
var d = Math.sqrt(dx * dx + dy * dy);
if(d < ability.radius + player.radius){
```



The network structure

Real time networking problem



Websockets

-What's a WebSocket

- Protocol of communication over TCP.
- Much faster than HTTP + AJAX.
- New technology standardized by the W3C
- Lack of server support

Socket.IO

-Whats Socket.IO

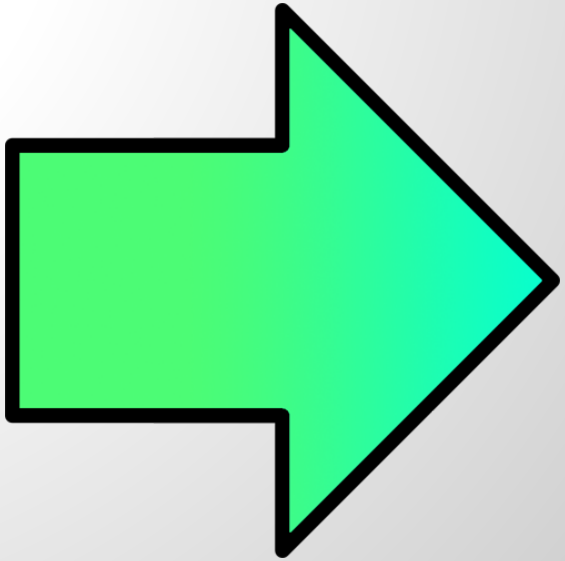
- WebSocket Wrapper
- Easy to use
- Downgrades

Node.js

-What's Node.js

- V8's is FAST
- Client/server : javascript/JSON.
- Open source!

Demo - Socket.IO



Socket.IO Demo Explained

Client

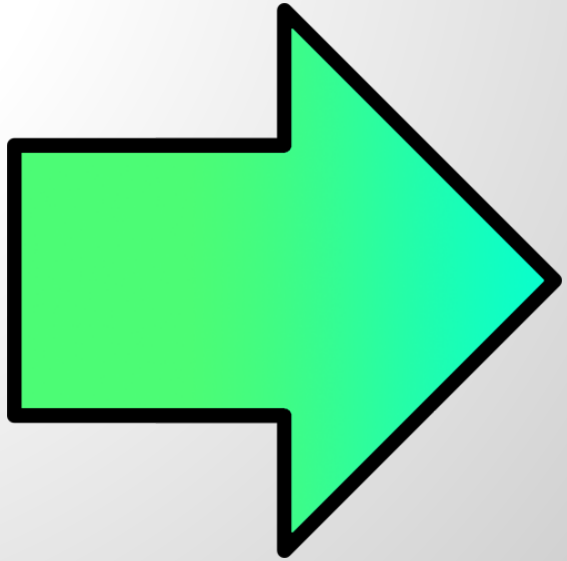
```
<head>
<title>socket io test!</title>
<script src="/socket.io/socket.io.js"></script>
<script>
  var socket = io.connect('http://localhost');
  window.onload = function(){
    var btn = document.getElementById('btn');
    var text = document.getElementById('text');

    btn.onclick = function(){
      addMessage('you ' + ": " + text.value);
      socket.emit('btnClick',{message: text.value});
      text.value = "";
    };
  };
  socket.on('receiveMessage',function(data){
    addMessage(data);
  });
  function addMessage(text){
    var message = document.createElement('div');
    message.innerHTML = text;
    document.body.appendChild(message);
  }
</script>
</head>
<body>
  <input type="text" id="text"/>
  <button id="btn">Send message</button>
</body>
```

Server

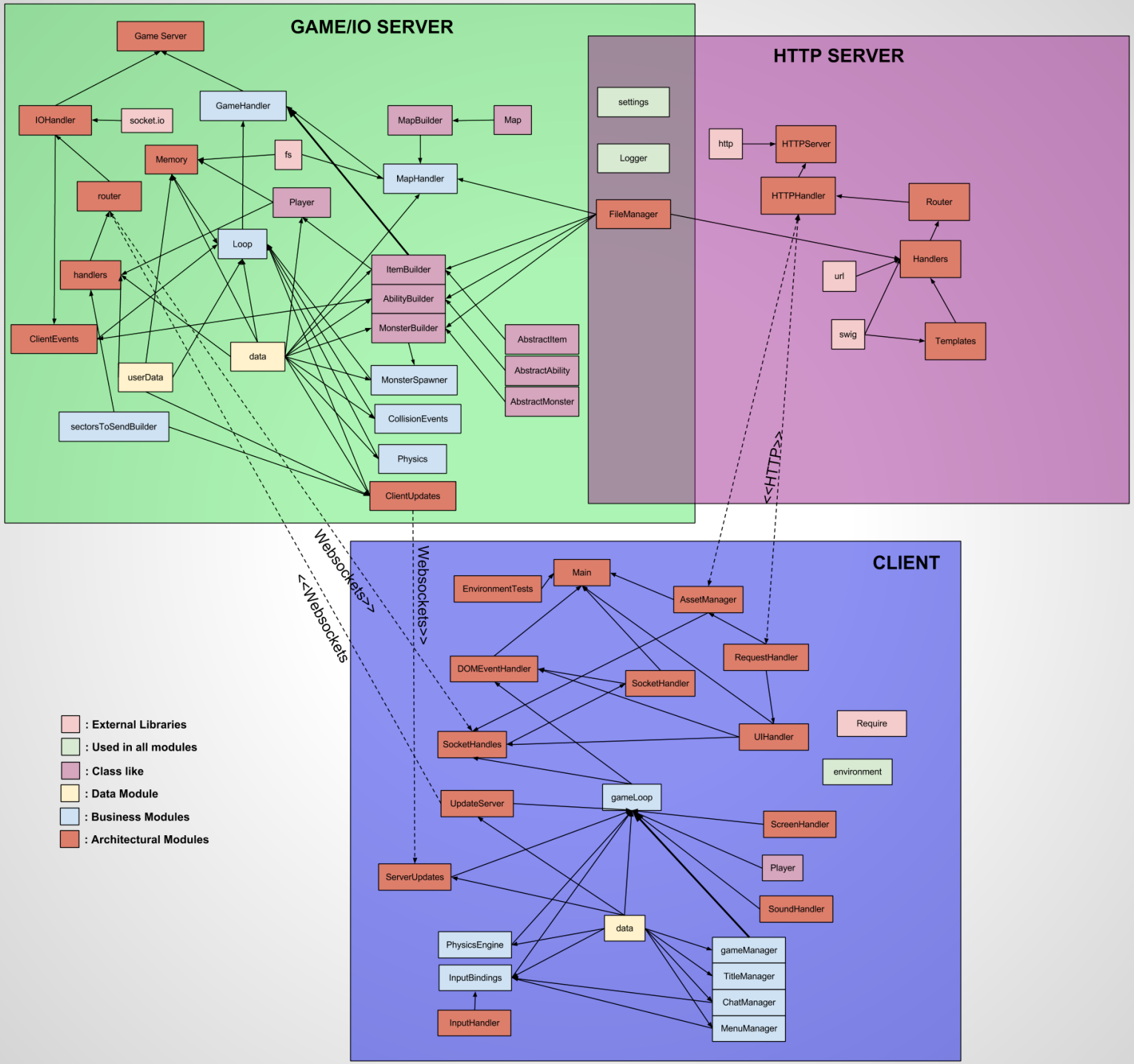
```
var app = require('http').createServer()
  , io = require('socket.io').listen(app)
  , fs = require('fs');

app.on('request',function (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);
  });
});

app.listen(80);
var nextUserId = 1;
//this is where the magic happens
io.sockets.on('connection', function (socket) {
  socket.name = "user" + nextUserId++;
  console.log(socket);
  socket.on('btnClick',function(data){
    console.log(data);
    socket.broadcast.emit('receiveMessage',socket.name + ": " + data.message);
  });
});
```

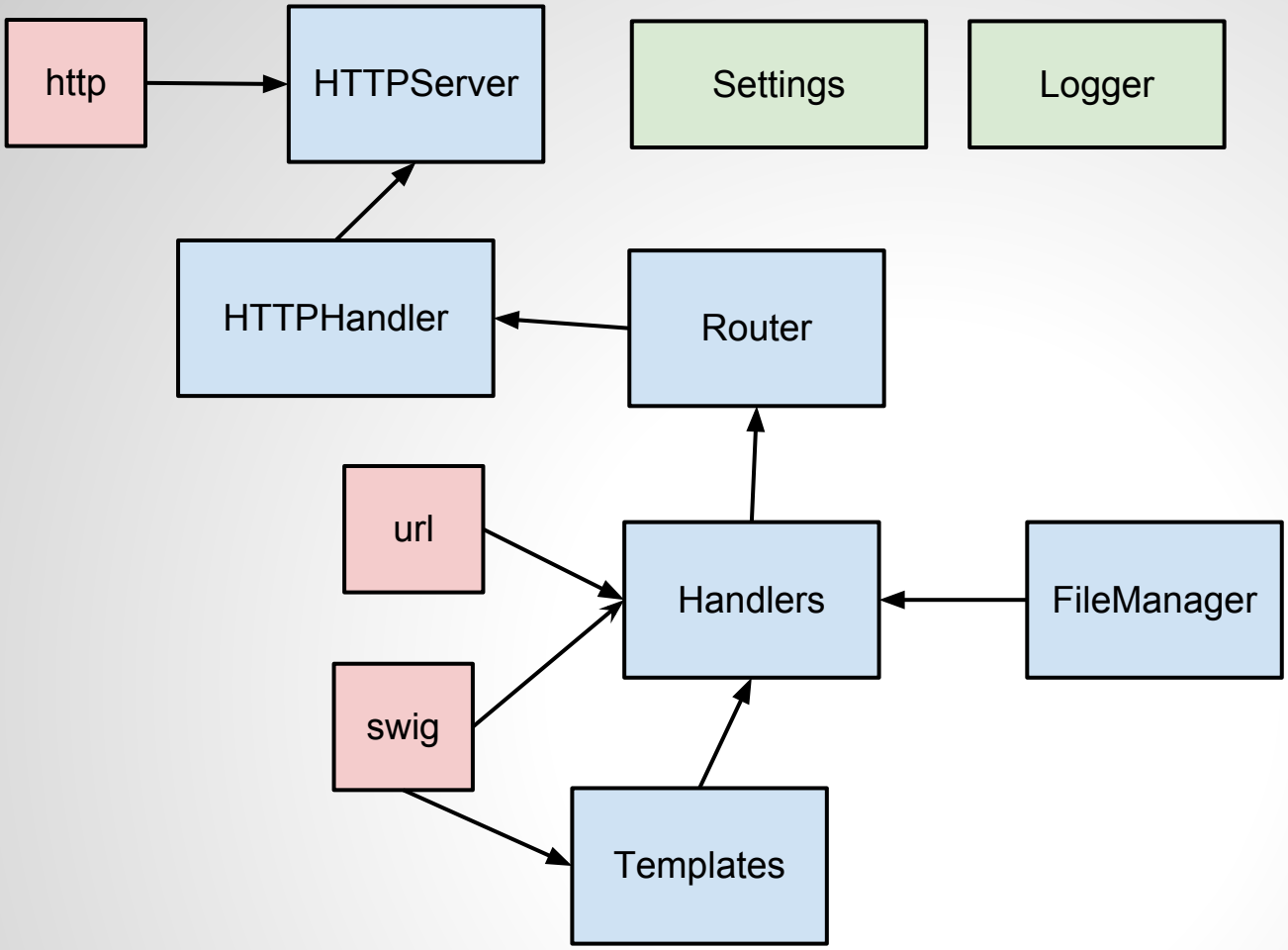
Demo - RUBBLE

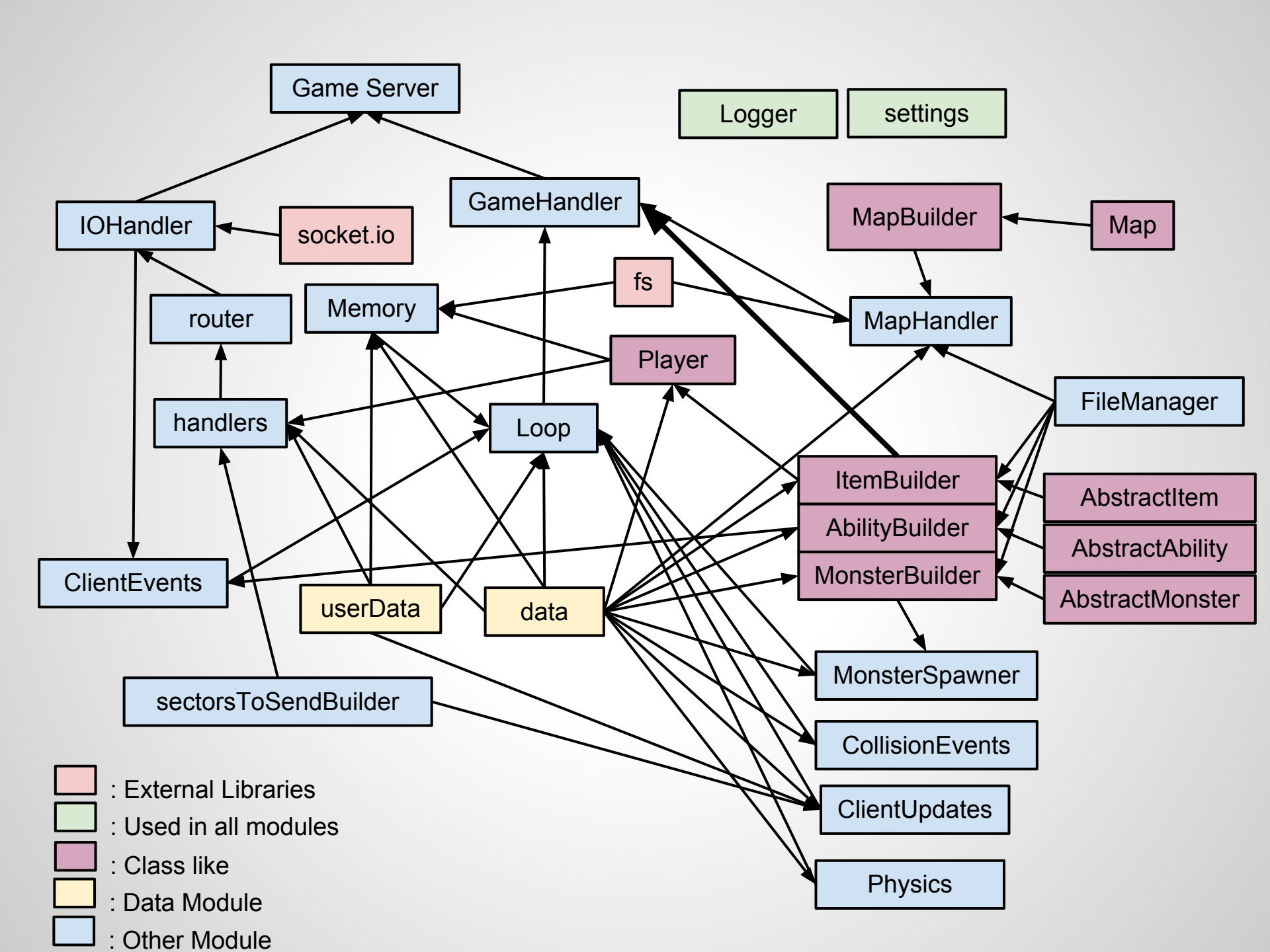
OUR PROJECT!



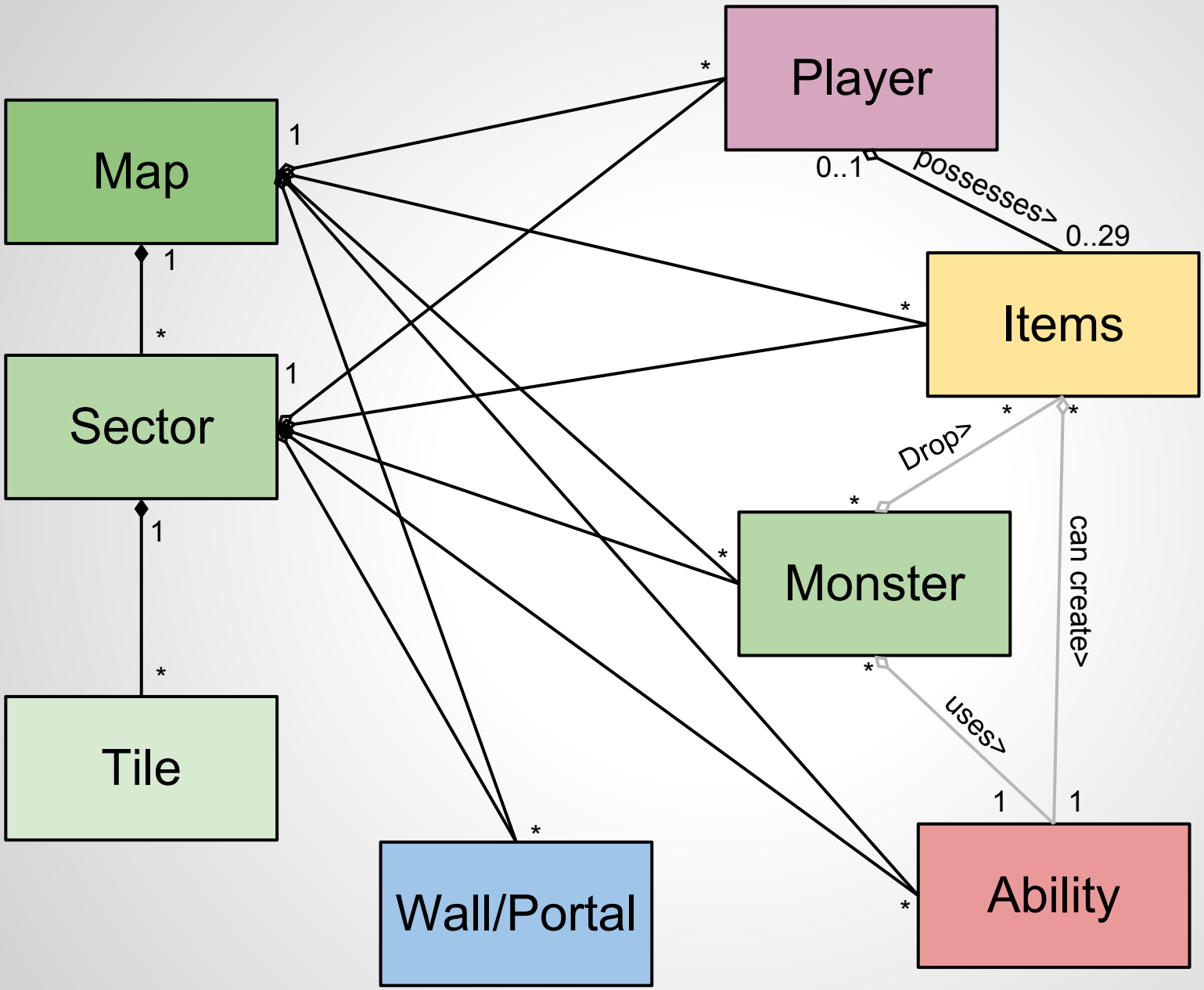
Module Diagram




http
HTTPServer
Settings
Logger
HTTPHandler
Router
url
Handlers
FileManager
swig
Templates
:external libraries
:used in all modules
:other module


Game Server
Logger
settings
GameHandler
IOHandler
socket.io
MapBuilder
Map
fs
router
Memory
MapHandler
Player
handlers
Loop
FileManager
ItemBuilder
AbstractItem
AbilityBuilder
AbstractAbility
MonsterBuilder
AbstractMonster
ClientEvents
userData
data
MonsterSpawner
sectorsToSendBuilder
CollisionEvents
ClientUpdates
Physics
: External Libraries
: Used in all modules
: Class like
: Data Module
: Other Module

Data structure ERD



Builder modules

-The Builder modules

- MonsterBuilder
- AbilityBuilder
- ItemBuilder

-Use shared JSON files

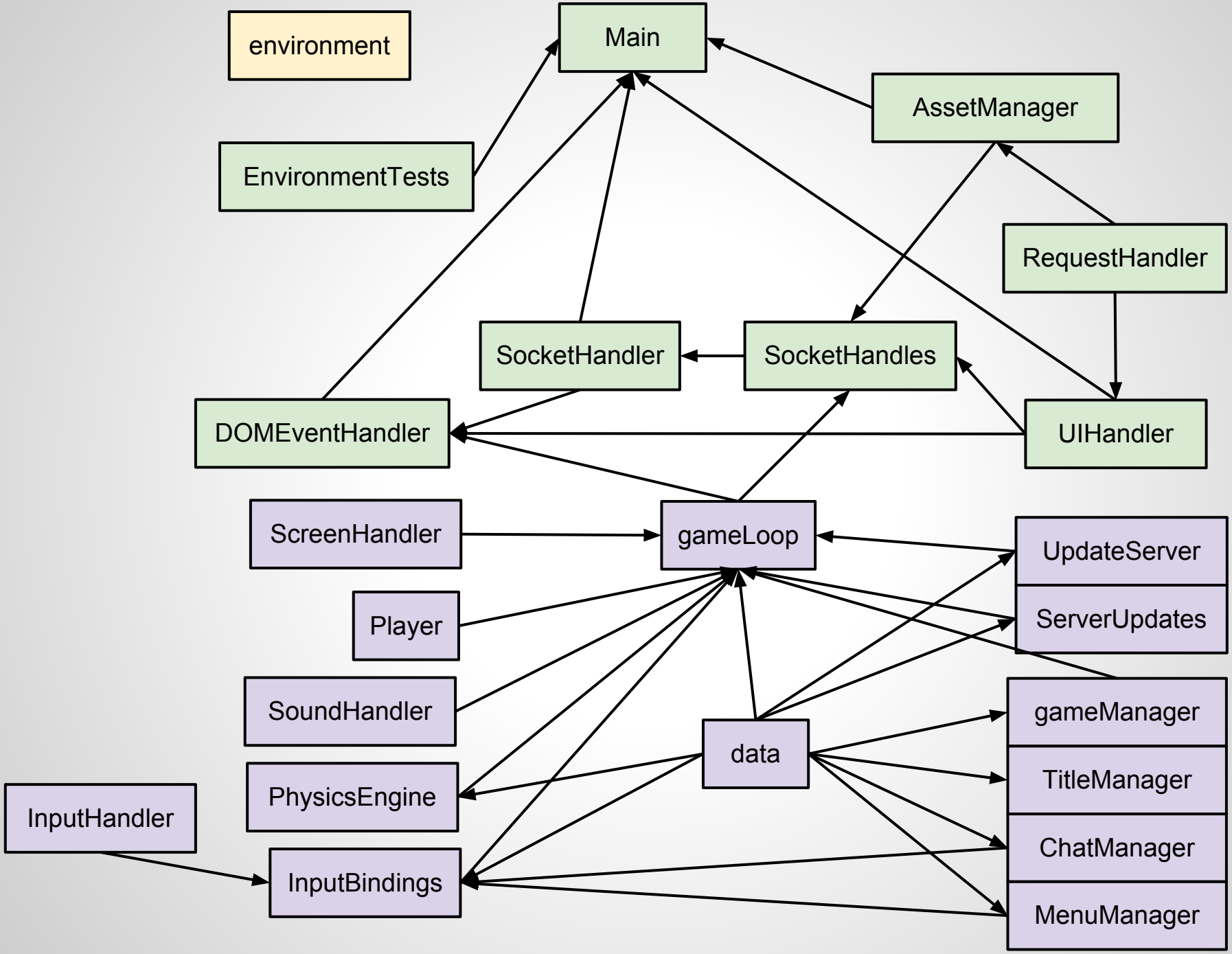
-Create constructor functions (= classes, kinda)

Actual Game Loop (server)

```
//Receive client events
clientEvents.update();

//Collision
var collisions = physics.update();
collisionEvents.update(collisions);

//Make sure all entities' sectors' are updated.
for(var mapName in data.maps) {
    data.maps[mapName].update();
}

//Spawn monsters
monsterSpawner.update();

//Store if necessary into backup file
memory.backup();

//update clients
clientUpdates.update();
```

Modules include:

- ClientEvents
- ClientUpdates
- Physics
- CollisionEvents
- MonsterSpawner
- Memory


environment
Main
AssetManager
EnvironmentTests
RequestHandler
SocketHandler
SocketHandles
DOMEventHandler
UIHandler
ScreenHandler
gameLoop
UpdateServer
Player
ServerUpdates
SoundHandler
gameManager
data
PhysicsEngine
TitleManager
InputHandler
ChatManager
InputBindings
MenuManager

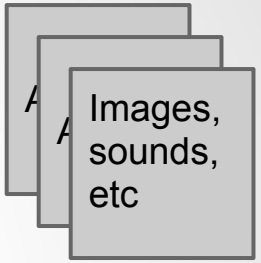
Actual Game Loop (Client)

```
//physics
gameData.player.doActions(inputBindings.keysPressed);
physicsEngine.update();
gameData.player.move();

//Play sounds
sounds.update();

//update title
titleManager.update(deltaTime);

//update gameManager
gameManager.update();

//update Chat
chatManager.update(deltaTime);

//drawing
requestAnimFrame(function(){
});

//update server with new client info
updateServer.update(socket);
```

Minimizing data over network

Sending Static Data once:

Assets:



Maps.json

- location of graphic tiles
- location of walls

Items.json

- Item Description
- Item graphic
- name
- ability
- bonus
- etc

Monsters.json

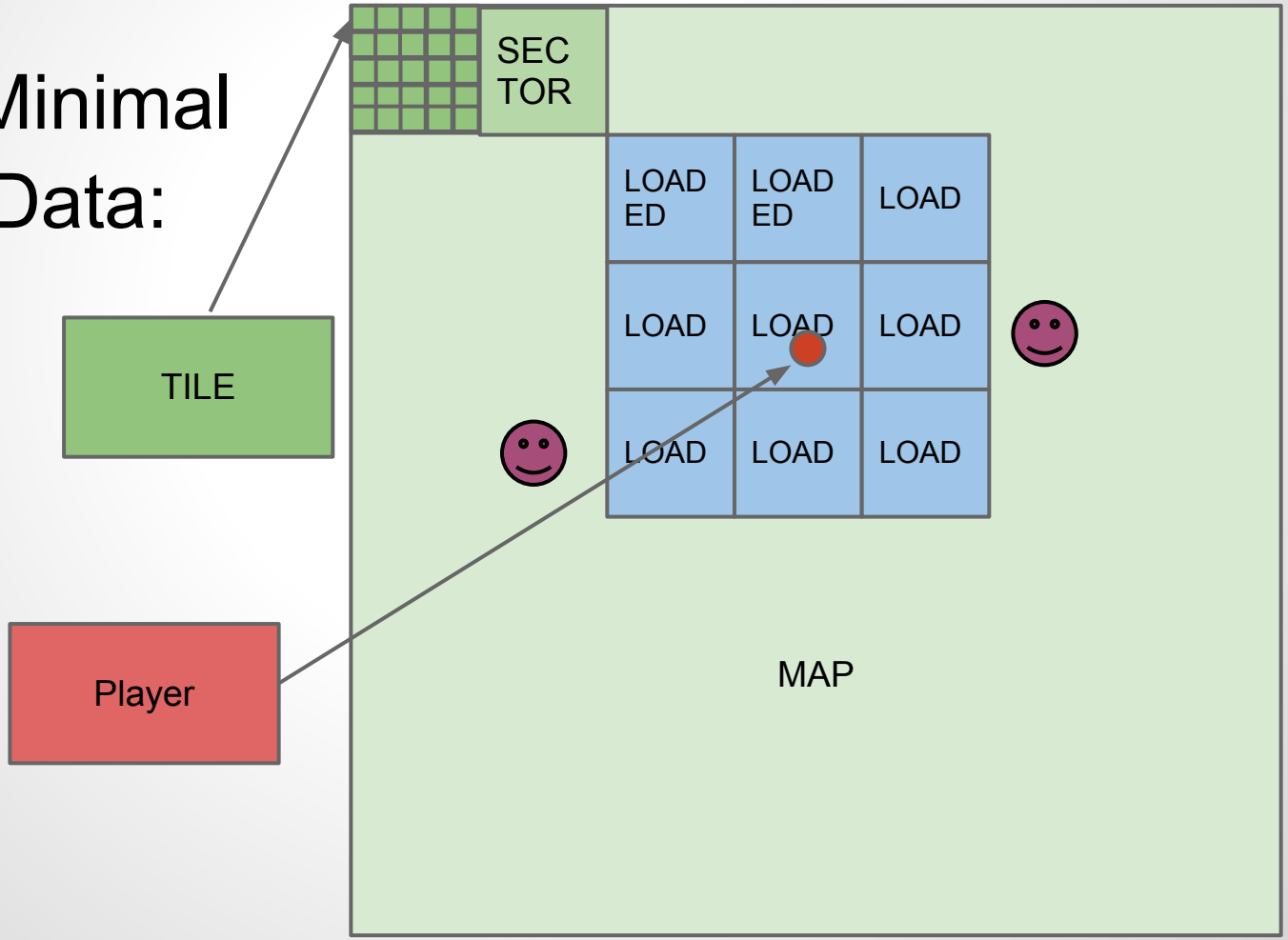
- monster graphic
- HP
- maxHP

Ability.json

- ability size
- ability color
- name
- damage

Minimizing data over network

Sending Minimal Dynamic Data:



Wrap-up

Software/libraries used

Libraries

-Client:

- require

-Server:

- swig
- socket.io

Software

- Tiled
- Chrome Dev Tools
- Sublime Text 2
- Google Drive
- BFXR
- Node.js (obviously)

Wrap-up

Teamwork

Wrap-up

What to do next?

- A functional donation system.
- Facebook/google login
- Being able to change a registered user's password
- Being able to register from a guest account
- Creating more complex abilities (shields, dash, etc)
- Ability to have friends
- Ability to create alliances
- Player/enemy animations
- Optimizing network communication
- Use some predictive algorithms for laggy users
- Much larger maps
- More control options(mouse/gamepad)
- Ability to use custom controls(key mapping)
- Item trading
- Duel arenas
- Banks

Thank You!

Questions?

...