ROE.Landmark is global namespace for map related and later I use “l.method()” with “l” as shortcut to it. Js namespace is designed as global, and means that only one map can work on a screen at one time.

The Atom of Map is what I call “Land”. It can be Landmark with river, mountain or Village of some type and each “Land” has coordinate in database. Each Land has size, it is always a square and depends on scale. Scale values now can be only - 1, 0.75, 0.5, 0.25. Initial size of Land is 84px, that is original size of all asset images. Current size of Land after scaling you can look into “l.landpx” means 84\*0.5=42. Also there is green background that I call “Earth”. It is also square - always 6 lands on 6 lands – 6x6. Scaled size of earth is in “l.earthpx”.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 | 6 |
| 2 | L | L | L | L | L |
| 3 | L | L | L | L | L |
| 4 | L | L | L | L | L |
| 5 | L | L | L | L | L |
| 6 | L | L | L | L | L |

As for html - Map is a DIV with positioned absolute Canvases for each Earth. Depending on scale one Earth/Canvas can be from 504px to 126px.

Left and Top of each Canvas are translated db cords – “x \*landpx % earthpx”.

So DIV is a viewport through which you look into map, and to move inside you need to change its:

-webkit-transform: translate(1023px, 619px);

This numbers are stored in “l.rx” and “l.ry”. To move viewport you can use “l.gotopx(rx, ry)”.

If you put translate(0px, 0px); than village(0,0) will be in upper left corner.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0,0 |  |  |  |  |  | Left: 504px, Top: 0px  Earth |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  | 0,0 |  |  |  |  |  |  |
|  |  | Earth |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  | Green is Viewport Mobile Screen |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |

Main cycle of map is:

start(x, y)

update() -> paint() -> fill() -> moredata() -> canvas.paint()

start() is very similar to paint method and differs from main cycle that it for sure knows that map is empty and need to be filled and has to have starting x, y center point.

**fill()** method creates canvases. Input parameter is a viewport zone left upper and right down. This coordinates are moduled at earthpx, so canvases are always in the same place in relation to div coordinates.

X

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 0 | 0 |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 6 | 0 |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 0 | 6 |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 6 | 6 |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 12 | 0 |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

Viewport

And fill() creates 6 canvases for it.

While creating canvases it also fill “l.earth” object. It consist of 2 indexes:  
- by string “left\_top” of canvas  
- by left and inside of it object of all top  
For sample on top:  
l.earth =  
{  
 c: { “0\_0”: true, “504\_0”: true, “1008\_0”: true, “0\_504”: true, “504\_504”: true, “1008\_504”: true },  
 x: { “0”: { “0”: true, “504”: true }, “504”: { “0”: true, “504”: true }, “1008”: { “0”: true, “504”: true } }  
}

While fill() executes it checks if earth already exist in an object and if not creates it.

All new created canvases are sent to moredata.

**moredata()** responding for loading all data from websql, localstorage and ajax.

Everything about Map stored in localStorage uses conception of “<KeyName>\_<RealmId>\_<PlayerId>” like “MapLandTypes\_39\_1185989”. As localStorage supports only strings, most of storages are JSON strings and at first time map executes moredata, they restored from JSON.

Js Api for this contains:  
BDA.Database.LocalSet(key, value)  
BDA.Database.LocalGet(key)  
BDA.Database.LocalDel(key) where key is “MapLandTypes”

ROE.Landmark

WebSql

localStorage

MapLandTypes

l.landtypes

l.players

MapPlayers

Landmarks

Villages

MapClans

l.clans

AJAX

One time

Each page load + Merge

l.land

l.villages

Loads once for needed canvases

MapVillArea

MapLandArea

l.landcoords

l.villcoords

If old, data updates

If exist, do not load from ajax, never changes

Details:

1. LandTypes – is cache of all mountains, rivers, trees parts. It cache one time into localStorage from ajax. While loaded once stored in “MapLandTypes\_<RealmId>\_<PlayerId>” and in “l. landtypes”

{"156":{"idt":1,"idtp":156,"image":"hills1BL.png"},"157":{"idt":1,"idtp":157,"image":"hills1BR.png"}, … }

1. Players and Clans – are stored in localStorage and restored each page reload. When Ajax asked for some earth to reload, it returns all Clans and Players that in this area. It is merged with current l.clans and l.players and resaved to localStorage with \_savelocalplcl(). This means localStorage growth all the time when you move through the map.
2. Landmarks and Villages – are stored in WebSql. While moredata asked for array of earth coordinates, function checks MapVillArea and MapLandArea if such landmarks were already once requested and load data from WebSql. WebSql used for 2 reasons – for offline access and because its quicker in general than Ajax. As \_savelocalplcl(), the \_savelocalvill() saves new information from Ajax to WebSql if area mentioned in MapVillArea is more than 5 min old.

**canvas.paint() –** responsible for draw info on canvas. Draw happened in stages:

1. Drawing of green background
2. Going through each of 6x6 land and check if there is village or landmark for it.
3. If village drawing, there are also highlights flags to draw, support/attack icons and consolidation images – which village is promoted and which absorbed.

Notes:

1. To draw it converts global db coordinates into local canvas coordinate (xx, yy)
2. All images ever loaded stored inside bigimgs object in form of { “<url>”: Image() }
3. preload() function returns Image from bigimgs or set new Image().onload event.

**select() –** village can be selected on a map with blue square, that gives you details on top and actions to do with it. When clicked on square second time it centering and executes a function \_**clearInvisibleCanvases**(). While developing a map, at first all loaded once canvases were on a page, until you go to a vov, that clears it. This approach gives crashes to mobile app after of some moving through a map. So we decided to remove unneeded canvases, but that also gives us crashes. After all we found that only way to clear memory without crashing is make canvases 1px width and height.

**info() –** executedon start() sets l.point\_changed = [function(x, y)]. When select() called, it calls pointdb() which checks if its other place village executes this event – point\_changed. Note: current selected village, or land x,y can be found at l.px, l.p.y. point\_changed is big, but in general it just show information from l.villages and ajax loading troops.

**zoom()** – reloads whole map in other scale value centering on the same village that was selected. Before that to scale we use “translate: scale(0.5)” but it was hard for devices, so now it is removing all old canvases (sets 1px width/height) and with new position rx, ry + scale + new landpx, earthpx - it creates map newly.

**checkvills () –** responsible for topicality of villages (not more than 5 minutes old). Check happens each 10 sec. villold() check that any of \_screenarea().villcoords is older than 5 minutes, then it executes update from ajax and refill().

**refill() -**   
 refill(true) – repaints all canvases  
 refill() – reload all canvases, which has villages that changed in some way, icon or flag  
Used by MapHighlights for example.

\_updatePaintCycle is another interval (1 sec) that was added because a bug of black parts of map sometimes. That black parts disappear after next small move and is connected to “l.nomorepaint” flag. It was added, to make moredata() sync. update() whenever asked tell true or false, if one of 4 sides of viewport is on black place, where there is no canvas. This check “if(update()) paint()” happens all the time someone moving through the map, so in this case l.nomorepaint means that no worries - loading is started, in the end of moredata() there is also check for update/paint. But if person stops moving a map in the middle of drawing canvases, and will be outside of zone that user requested, it will get black zone. For this \_updatePaintCycle updates each second.