Technical description	
1	**General**
10a	all track elements are instances of classes: TrainSignal, Track, Point, Block
10b	these instances own static and dynamic properties (variables)
10c	the static properties contain mainly geographic informations (position, neigbours etc.)
10d	the dynamic property 'ok' (not faulty) is used in all classes of track elements
10e	tracks and points own always a property 'free' (not occupied)
10f	speciality: all instances of tracks and points own the properties 'ZfL' and 'ZfR' (train route is set in left resp. in right direction)
10g	all instances of tracks and points own the properties 'olL', 'olR' and 'flProt': overlap element for right/left direction and element of a flank protection
11a	The train routes are set with the geographic principle: starting from the target signal backwards, points tied in both positions, until the start signal is found.
11b	With setting the train route, also the overlap and the flanc protection elements ars searched and set
11c	a track element (track or point) belayed with a train route ('ZfR' or 'ZfL') is reset, if it was occupied and then free and if the preceding track element is free and not belayed with a train route
2	**Software**
2a	program language: Java Script with canvas
3	**Files**
3a	root: index.html calls the canvas, sets the buttons and calls the 3 js scripts
3b	*/scripts/...classes.js* defines the classes (see also file *classes.pdf*)
3c	*/scripts/...functions.js* defines all functions
3d	*/scripts/...definitions.js* defines all parameters
4	**Classes**
4a	TrainSignal
4b	Track
4c	Block
4d	Point
5	**functions**
	completeTopo() sets all neighbours L and various other redundant data
	onNewLoad() resets all dynamic parameters [Grundzustand] and redraws everything in magenta
	initialize() - if all docs loaded - starts loadDoc for the different tables, normalizes all elements and redraws everything
	operationInstruction()
	drawStatics() draws the title (here: "F-Dorf" in a rectangle)
	showFailureText()

	hideFailureText()
	failureTextResetClick()
	onClickCanvas(event) checks the coordinates of the mouse and calls with them all methods named ...click() of all instances of the classes
	flashGeneric()
	onMouseDownCanvas(event)
	onMouseDownMoveConvas(event)
	onMouseUpCanvas(event)
	reDraw()
	findRouteBackwards()
	preBuildTrainRoute()
	setPointsForTrainRoute()
	startBuildingTR()
	buildTrainRoute()
6	**setting a train route (Zf)**
6a	When a TrainSignal is clicked and it is a possible start signal, with mouse down moving a yellow line is drawn
6b	when then the mouse is over a possible target signal, the method *findRouteBackwards()* is activated; if ok, the signal changes to yellow
6c	when then releasing the mouse, the method *preBuildTrainRoute()* is activated; if ok, the route elements are drawn in yellow
6d	if ok (*preBuildTRok* = TRUE), the function *setPointsForTrainRoute()* is activated, the switching points begin flashing for the time *tChangeSwitch*
6e	afterwards, the function *startBuildingTrainRoute()* is started
6f	the function *buildTrainRoute()* is started: here also the flanc protection and the overlap are defined and set
6g	by succesive occupying and setting free the route elements, the train route is released
7	**block** (see also register *block*)
7a	the simulation of the blocks is simplified to avoid actions on the neighbour stations
7b	if a section track (Streckengleis) gets occupied, the corresponding block changes to the state "blocked", direction to F-Dorf
7c	to release the block, a train route has to be set and a train has to enter the station (F-Dorf)
7d	departure train route from F-Dorf: by setting a route the block changes to the state "preblocked"
7e	when the corresponding train get on the section track, the block changes to the state "blocked"

7f	when after this the corresponding section track gets free, the block changes to the state "free" [not as in reality!]