



CORFU ESCAPE ADVENTURE

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ



ΕΤΟΣ : 2019-2020

*ΙΟΝΙΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ
ΦΟΙΤΗΤΡΙΑ : ΠΑΠΑΠΟΣΤΟΛΟΥ ΜΥΡΤΩ ΑΝΑΣΤΑΣΙΑ Π2016030
ΕΠΙΒΛΕΠΩΝ ΚΑΘΗΓΗΤΗΣ : ΔΗΜΗΤΡΙΟΣ ΡΙΓΓΑΣ*

ΠΡΟΛΟΓΟΣ

Σκοπός της εργασίας αυτής είναι η ανάδειξη των περιοχών ιδιαίτερου ενδιαφέροντος της πόλης της Κέρκυρας , η διάδοση πληροφοριών για ιστορικά σημεία και η καλύτερη διάχυση ιστορικών γνώσεων για αυτά , στον συμμετέχοντα της εφαρμογής μέσα από νέες τεχνολογίες σε συνδυασμό ψυχαγωγικών διαδικασιών, όπως τα παιχνίδια πεδίου και προσανατολισμού.

Καθώς εξελίσσονται και εμφανίζονται νέες και πιο ισχυρές κινητές συσκευές, όπως “smartphone” και “tablet”, εμφανίζονται και νέοι τρόποι αλληλεπίδρασης για εξερεύνηση μέσα από αυτές από τους χρήστες και προγραμματιστές. Οι νέες τεχνικές πληροφόρησης σε συνδυασμό με την ψυχαγωγία, μέσα από τις συσκευές, οι οποίες χρησιμοποιούν ένα ευρύ φάσμα εισόδων και αισθητήρων, όπως ηλεκτρονικές πυξίδες, αισθητήρας GPS και άλλων, επιτρέπουν την δημιουργία εξατομικευμένων εμπειριών. Τέλος παρουσιάζουμε, μια εφαρμογή η οποία αναδεικνύει τις ικανότητες και τις δεξιότητες του προγραμματιστή στην ανάπτυξη παιχνιδιών τοποθεσίας και διάδοσης πληροφορίας με ψυχαγωγικό τρόπο.

Θα ήθελα να ευχαριστήσω όσους συνέβαλαν στην ολοκλήρωση της εργασίας αυτής.

Λέξεις κλειδιά: εκπαιδευτικά παιχνίδια, ψυχαγωγία, ιστορικά μνημεία

ΕΙΣΑΓΩΓΗ

Πριν αρκετά χρόνια ο άνθρωπος ανακάλυψε τους ηλεκτρονικούς υπολογιστές και από τότε χρησιμοποιούνται συνεχώς. Αυτή η τεχνολογία έχει διαμορφώσει την εξέλιξη της ανθρωπότητας και καθορίζει την ζωή των ανθρώπων με αρνητικό και θετικό τρόπο.

Σε αυτή την εργασία θα αφοσιωθούμε μόνο στη θετική επίδραση της τεχνολογίας αυτής με σκοπό την προσπάθεια δημιουργίας ενός παιχνιδιού που συνδυάζει εκπαίδευση και ψυχαγωγία.

Στις μέρες μας, είναι πλέον γνωστό πως τα ψηφιακά μέσα έχουν γίνει μέρος της καθημερινότητάς μας και ιδιαίτερα των παιδιών και των νέων (Chan & Rabinowitz, 2006; Whitton, 2007; Zhang, Liu, Wang, & Piao, 2010). Δηλαδή, υπάρχει επιθυμία για μάθηση με αλληλεπίδραση που θα κινητοποιήσει και θα αυξάνει το ενδιαφέρον τους (Tapscott, 1998). Σύμφωνα με τον Johnson το 2005, οι περισσότεροι κάλυπταν την ανάγκη για μάθηση με τον παραδοσιακό τρόπο. Ωστόσο, επειδή το παιχνίδι υπάρχει στην ζωή μας και ο κόσμος αφιερώνει αμέτρητες ώρες σε αυτό και χρήματα (Κουφού, 2015) μπορούμε μέσα από αυτό ταυτόχρονα να μάθουμε αλλά και να διασκεδάσουμε.

Έτσι, λέμε πως το ηλεκτρονικό παιχνίδι αποτελείται από μία ηλεκτρική συσκευή και είναι ένα απλό παιχνίδι προσομοιωμένο σε λογισμικό (Galloway, 2006). Για αυτό τα τελευταία χρόνια, ο χώρος των ηλεκτρονικών παιχνιδιών αποτέλεσε πεδίο έρευνας με στόχο την αξιοποίηση των εμπορικών παιχνιδιών στην εκπαιδευτική διαδικασία, καθώς και το σχεδιασμό εκπαιδευτικών παιχνιδιών. Οι Kirriemuir και McFarlane (2004) συμπληρώνουν ότι τα ψηφιακά παιχνίδια παρέχουν οπτικές ψηφιακές πληροφορίες σε έναν ή περισσότερους παίκτες, επιτρέπουν την εισαγωγή δεδομένων από τους χρήστες, διαχειρίζονται τα δεδομένα αυτά σύμφωνα με ορισμένους προγραμματισμένους, για το παιχνίδι, κανόνες και τελικά προβάλλουν τις ψηφιακές πληροφορίες στους παίκτες.

Τα ηλεκτρονικά παιχνίδια / εφαρμογές έχουν αρκετές κατηγορίες όπως, παιχνίδια περιπέτειας (ακολουθούν μια ιστορία, περιλαμβάνουν προβλήματα για επίλυση, γρίφους), λογικής (στα οποία ο παίκτης πρέπει να αντιδρά πιο γρήγορα ώστε να πετύχει π.χ. μια μαθηματική εξίσωση), online (τα οποία ο παίκτης μπορεί να παίξει διαδικτυακά και με άλλους παίκτες), serious games και παιχνίδια με βάση τοποθεσίας. Εκτιμάται λοιπόν ότι προκειμένου να ανακαλύψουμε πώς μπορούμε να σχεδιάσουμε ελκυστικά εκπαιδευτικά παιχνίδια θα πρέπει να κατανοήσουμε τι είναι αυτό που κινητοποιεί τους ανθρώπους να παίζουν τα “serious games” και τα “location based games” με τα οποία και θα ασχοληθούμε στην εργασία αυτή.

Τα serious games είναι παιχνίδια, που ο κύριος σκοπός τους είναι να παρέχουν πληροφορίες, να εκπαιδεύουν και να ευαισθητοποιήσουν για μια ιδέα ή άποψη πέρα από την διασκέδαση και την ψυχαγωγία.

Τα παιχνίδια βάσει τοποθεσίας (Location Based Games) είναι παιχνίδια που παίζονται στον πραγματικό κόσμο χρησιμοποιώντας συσκευές χειρός, όπως κινητά τηλέφωνα ή PDA. Λαμβάνοντας παιχνίδια σε αστικούς δρόμους, αυτά τα παιχνίδια χρησιμοποιούν τεχνολογίες SMS και GPS για να επιτρέψουν στους παίκτες να συμμετάσχουν στο αστικό τοπίο με διαφορετικό τρόπο.

Σκοπός αυτής της εργασίας είναι η μάθηση του τι εστί εκπαιδευτικό παιχνίδι σε θεωρητικό και πρακτικό κομμάτι. Δηλαδή, εμπέδωση και κατανόηση τι είναι ένα ηλεκτρονικό παιχνίδι, η κατηγορία serious

game, location based games και κατόπιν η εκμάθηση των εργαλείων που θα χρησιμοποιηθούν και τα στάδια κατασκευής του.

Η μεθοδολογία προσέγγισης του προβλήματος αυτού θα είναι η κατασκευή ενός αρχικού σταδίου ενός ηλεκτρονικού εκπαιδευτικού παιχνιδιού και παράλληλα η εκμάθηση του τρόπου σχεδίασης του τόσο στο θεωρητικό όσο και στο πρακτικό του σκέλος.

Τα κεφάλαια που θα ακολουθήσουν χωρίζονται σε έξι ενότητες, «θεωρία», «σχεδίαση», «υλοποίηση», «προβλήματα & λύσεις», «συμπεράσματα» και «βιβλιογραφία».

Στην πρώτη ενότητα θα δούμε το θεωρητικό σκέλος της εργασίας αυτής. Θα μελετηθεί το τί είναι ένα ηλεκτρονικό παιχνίδι, πως επηρεάζει στην μάθηση και θα αναλυθούν οι δύο σημαντικές κατηγορίες στις οποίες θα καταταχθεί η εφαρμογή μας.

Στην ενότητα «σχεδίαση», περιγράφεται η ιδέα της εφαρμογής, ο σκοπός και οι λειτουργίες. Εκτός αυτών υπάρχει και η αξιολόγηση από χρήστες που δοκίμασαν το προσχέδιο της εφαρμογής μας.

Έπειτα ακολουθεί η «υλοποίηση», στην οποία υπάρχει εξήγηση των γλωσσών προγραμματισμού, το περιβάλλον εργασίας και τα εργαλεία που θα χρησιμοποιηθούν. Θα εξηγηθεί βήμα ο κώδικας και τα στάδια ανάπτυξης του παιχνιδιού.

Στο κεφάλαιο «προβλήματα & λύσεις», διευκρινίζονται τα προβλήματα που εμφανίστηκαν κατά την διάρκεια δημιουργίας της εφαρμογής και πώς αντιμετωπίστηκαν.

Στον επίλογο της εργασίας θα αναφερθούν τα συμπεράσματα και η βιβλιογραφία.

ΠΙΚΑΝΑΣ ΠΕΡΙΕΧΟΜΕΝΩΝ

ΘΕΩΡΙΑ	5 - 13
Ηλεκτρονική Μάθηση	5 - 6
Ηλεκτρονικά Παιχνίδια	7 - 8
Κατηγορίες Ηλεκτρονικών. Παιχνιδιών	9 - 13
ΣΧΕΔΙΑΣΗ	13 - 29
Αξιολόγηση	14 - 18
Εργαλεία Υλοποίησης	18 - 29
ΥΛΟΠΟΙΗΣΗ	30 - 76
React	30 - 33
Βήματα Υλοποίησης	33 - 70
Αξιολόγηση	71 - 76
ΠΡΟΒΛΗΜΑΤΑ & ΛΥΣΕΙΣ	77 - 81
ΣΥΜΠΕΡΑΣΜΑΤΑ	82 - 83
ΒΙΒΛΙΟΓΡΑΦΙΑ	84 - 89

Η ραγδαία ανάπτυξη των Τεχνολογιών της Πληροφορίας και των Επικοινωνιών (Τ.Π.Ε.), του Διαδικτύου, καθώς και των κινητών και «έξυπνων» συσκευών έχουν εισβάλλει δυναμικά στη ζωή μας έχοντας μετασχηματίσει τον τρόπο με τον οποίο οι άνθρωποι επικοινωνούν και εκπαιδεύονται. Οι σύγχρονες ανάγκες για μεγαλύτερη αυτονομία και ευελιξία ως προς τον τόπο, το χρόνο και το ρυθμό μάθησης έχουν οδηγήσει σε έναν καινοτόμο τύπο διδασκαλίας και μάθησης, την ηλεκτρονική μάθηση (e-learning).

Ηλεκτρονική Μάθηση

Η ηλεκτρονική εκπαίδευση και μάθηση είναι ένα αναπτυσσόμενο φαινόμενο που συνεχώς ερευνάται και εξελίσσεται μαζί με τα τεχνολογικά μέσα. Ο συγκεκριμένος τρόπος μάθησης τυγχάνει όλο και μεγαλύτερης αποδοχής από την επιστημονική κοινότητα, τους εκπαιδευτικούς, τους μαθητές αλλά ταυτόχρονα υπήρξε και ερευνητικό ενδιαφέρον για ανάπτυξη της τεχνολογικής μάθησης.

Αρχικά, ηλεκτρονική μάθηση (e-Learning) εννοείται η χρήση των νέων τεχνολογιών πολυμέσων και του διαδικτύου για τη βελτίωση της ποιότητας της εκπαίδευσης, μέσω της διευκόλυνσης του τρόπου πρόσβασης σε πηγές πληροφοριών και υπηρεσίες, αλλά και στην από απόσταση συνεργασία και ανταλλαγή απόψεων μεταξύ εκπαιδευτών και εκπαιδευόμενων, καθώς και των εκπαιδευόμενων μεταξύ τους. Οι Urban και Weggen (2000) ορίζουν αυτή την μάθηση σαν την παράδοση περιεχομένου μέσω ηλεκτρονικών μέσων, όπως το ίντερνετ, οπτικοακουστικά μέσα (ήχοι, βίντεο) και CD-ROM. Αντίστοιχα, χαρακτηρίζεται και ως η απόκτηση και χρήση της γνώσης που διανέμεται και διευκολύνεται με ηλεκτρονικά μέσα (Wentling, et al.,2000).

Επιπλέον, η ηλεκτρονική μάθηση ορίζεται και ως η συστηματική χρήση υπολογιστών σε δίκτυα, η οποία βελτιώνει την μάθηση, συνδέει ανθρώπους και πηγές πληροφοριών που υποστηρίζουν τις ανάγκες των χρηστών (Goodyear,2002).

Η ηλεκτρονική μάθηση διακρίνεται σε σύγχρονη και ασύγχρονη μάθηση. Σύγχρονη μάθηση ή τηλεεκπαίδευση (TechnologyBasedTraining) απαιτεί την ταυτόχρονη συμμετοχή και αλληλεπίδραση όλων των εκπαιδευόμενων και των διδασκόντων σε «πραγματικό χρόνο», με ήχο και εικόνα. Η ασύγχρονη μάθηση (web based learning), δεν απαιτεί την ταυτόχρονη συμμετοχή των εκπαιδευόμενων και των εισηγητών αλλά ούτε και να βρίσκονται συγκεντρωμένοι μαζί στον ίδιο χώρο ή την ίδια χρονική στιγμή. Μπορούν να επιλέγουν μόνοι τους το προσωπικό τους εκπαιδευτικό χρονικό πλαίσιο και να συλλέγουν το εκπαιδευτικό υλικό σύμφωνα με αυτό.

Επειδή έχει εξελιχθεί η μάθηση με την χρήση των νέων τεχνολογιών έχουν δημιουργηθεί καινούργια μοντέλα e-Learning.

Εφόσον, το e-Learning αποτελεί εξέλιξη της ασύγχρονης μάθησης και σύμφωνα με τους Connolly και Stansfield (2006), βασισμένοι στην άποψη του Taylor περί ηλεκτρονική μάθησης προτείνουν ένα μοντέλο τριών γενεών,

1^η γενιά. “The objectivist eLearning Model”, μοντέλο που χαρακτηρίζεται από το διαδίκτυο και το εκπαιδευτικό υλικό μεταφέρεται από φυσική μορφή σε ηλεκτρονική.

2^η γενιά. “The Flexible eLearning Model”, μοντέλο που χρησιμοποιεί προηγούμενες τεχνολογίες και παρέχει υλικό με διαδραστικό περιεχόμενο (πχ. Εικονικά μαθησιακά περιβάλλοντα).

3^η γενιά. “The Constructivist eLearning Model”, μοντέλο που είναι συνεργατικό και προωθεί την μάθηση μέσω εργαλείων όπως τα blogs, wikis, online κοινότητες, προσομοιώσεις και παιχνίδια.

Επίσης παρέχει την δυνατότητα ανάπτυξης συστημάτων για εκπαίδευση των χρηστών μέσω κινητών τηλεφώνων και τα λεγόμενα “έξυπνα” τηλέφωνα (smartphones).

Η συγκεκριμένη διαπίστωση επιβεβαιώνεται από διάφορα στατιστικά στοιχεία και συγκεκριμένα, οι Barodiya, Kushwah και Kaurav (2016) ισχυρίζονται ότι το 1990 το ποσοστό του εκπαιδευτικού συστήματος που εφάρμοζε τέτοιου είδους μάθηση ήταν στο 8% ενώ το 2014 έφτασε το 80%.

Από τα παραπάνω στοιχεία μπορεί να συμπεράνει κάποιος τη σημαντικότητα που αποδίδεται στον τομέα αυτό της εκπαίδευσης και πώς οι επιστήμονες συνεχώς την μελετούν και προσπαθούν να την εξελίσουν.

Από τις μαθησιακές στρατηγικές με χρήση νέας τεχνολογίας υπάρχουν αρκετές αλλά μόνο τρεις θα βοηθήσουν για την κατασκευή της εφαρμογής μας. Η πρώτη είναι η παιδαγωγική - μαθησιακή στρατηγική (pedagogical strategy) η οποία , εστιάζει στην εφαρμογή κατάλληλων διδακτικών στρατηγικών και μέσων, σχεδιασμένα με τέτοιο τρόπο, ώστε να επιτυγχάνεται η αποτελεσματική μάθηση. Στα πλαίσια της εντάσσονται οι μέθοδοι και οι διαδικασίες μάθησης των εκπαιδευομένων (ομάδες εκπαιδευομένων, συζητήσεις, forums, σεμινάρια, χρήση μαθηματικών κλπ.).

Η επόμενη είναι η ενεργητική μάθηση (active learning). Η συγκεκριμένη μέθοδος βοηθάει τον εκπαιδευόμενο να μάθει μέσω μίας εμπειρίας, μιας αλληλεπίδρασης , χωρίς την βοήθεια του εκπαιδευτή. Η τρίτη και τελευταία στρατηγική είναι η μάθηση διαμέσου επίλυσης προβλημάτων (problem solving learning - PBL). Η συγκεκριμένη μάθηση οδηγεί τον εκπαιδευόμενο στην επίλυση ενός προβλήματος με μια έννοια την οποία πρέπει να μελετήσει και να «μάθει»(π.χ. ένα κουίζ). Η μέθοδος PBL βρίσκεται σε «ταίριασμα» με τα νέα μοντέλα διδασκαλίας.

Εκτός των τριών γενεών του eLearning , υπάρχει και μία διάκριση μεταξύ των τριών διαφορετικών εννοιών: ηλεκτρονικής μάθησης (e-learning), μικτής μάθησης (blended learning) και κινητής μάθησης (mobile learning). έννοιες οι οποίες εξαρτώνται όχι μόνο από τις προόδους που σημειώνονται στον τομέα των τηλεπικοινωνιών, αλλά και από τους μετασχηματισμούς στους τομείς της διδασκαλίας και μάθησης. Στην παρούσα εργασία δίνεται έμφαση στην μικτή μάθηση και σε αυτή που ανήκει στην Τρίτη γενιά. Δηλαδή μάθηση με χρήση τεχνολογικών μέσων εξ αποστάσεως που έχουν πρόσβαση στο διαδίκτυο σε μορφή ηλεκτρονικού παιχνιδιού.

Ηλεκτρονικά παιχνίδια

Για να γίνει η σωστή χρήση της θεωρητικής ηλεκτρονικής μάθησης ερχόμαστε στην πρακτική εφαρμογή της που είναι ο σχεδιασμός εκπαιδευτικών μέσων.

Η ανάγκη ενός αποτελεσματικού διδακτικού μοντέλου που να μπορεί να προβλέπει και να καλύπτει τις ανάγκες του χρήστη, επιτρέποντας όμως και τον απαραίτητο έλεγχο από τους συντηρητές του συστήματος καθώς και την ενημέρωση και τις τυχόν διορθώσεις από τον εκπαιδευτή, κρίνεται απαραίτητη για την επιτυχία της επικοινωνίας και της εκπαίδευσης από απόσταση.

Ο ίδιος ο όρος της Παιχνιδοποίησης μας παροτρύνει να εξετάσουμε αρχικά το τι εννοούμε με τον όρο παιχνίδι και συγκεκριμένα ποια είναι τα ιδιαίτερα χαρακτηριστικά των παιχνιδιών. Σύμφωνα με τον Roger Caillois (1961) ο όρος παιχνίδι, έχει δύο διαστάσεις: “paidia” και “ludus”. Με τις διαστάσεις αυτές αναφερόμαστε στα ελεύθερα παιχνίδια όπως αυτό των μικρών παιδιών, και με τον όρο “ludus”, αναφερόμαστε στο παιχνίδι που έχει συγκεκριμένους κανόνες για επίτευξη συγκεκριμένων στόχων. Όπως αναφέρεται στο Bäck, 2008 τα παιχνίδια έχουν συγκεκριμένο στόχο (Pre-lusory goal), διέπονται από συγκεκριμένους κανόνες (Constitutive rules) και οι παίκτες αποδέχονται εθελοντικά αυτούς τους κανόνες προκειμένου να παίξουν αυτό το παιχνίδι.

Ο όρος playful design (Παιγνιώδης σχεδιασμός) αναφέρεται στο σχεδιασμό ο οποίος δίνει λιγότερη βαρύτητα στους κανόνες και εστιάζεται περισσότερο στο να προσφέρει μία παιγνιώδη εμπειρία, όπως αυτή προκύπτει από δραστηριότητες που εντάσσονται στον όρο “paidia” (Egg, 2013).

Δεν υπάρχει μία ακριβής κατηγορία ηλεκτρονικών παιχνιδιών καθώς διαφέρουν ανάλογα με το είδος. Τα βασικά είδη που υπάρχουν είναι τα computer games(παιχνίδια σε υπολογιστή), τα βιντεοπαιχνίδια, τα παιχνίδια κοινότητας (MassiveMultiplayerOnlineGameS και MassiveMultiplayerOnlineRolePlayingGames), τα mobile games(παιχνίδια σε smartphones)και αυτά των εικονικών κόσμων (Virtual Worlds). Όμως για αυτά τα είδη υπάρχουν μετά κατηγορίες παιχνιδιών όπως, παιχνίδια περιπέτειας (ακολουθούν μια ιστορία, περιλαμβάνουν προβλήματα για επίλυση, γρίφους), λογικής (στα οποία ο παίκτης πρέπει να αντιδρά πιο γρήγορα ώστε να πετύχει π.χ. μια μαθηματική εξίσωση), online (τα οποία ο παίκτης μπορεί να παίξει διαδικτυακά και με άλλους παίκτες) , serious games και παιχνίδια με βάση τοποθεσίας.

Ένα σημαντικό χαρακτηριστικό των παιχνιδιών είναι ότι η ενασχόληση μαζί τους δεν είναι υποχρεωτική, είναι μία προσωπική εκούσια απόφαση. Δεύτερο χαρακτηριστικό των παιχνιδιών είναι ότι δίνουν στους παίκτες επιλογές. Σε όλη τη διάρκεια του παιχνιδιού οι παίκτες κάνουν τις προσωπικές τους επιλογές, όπως με ποια σειρά να κινηθούν μέσα στο παιχνίδι, οι οποίες καταλήγουν σε συγκεκριμένα αποτελέσματα. Αυτή η ιδιότητα των παιχνιδιών δίνει στους παίκτες μία αίσθηση ελέγχου και αυτονομίας.

Επίσης, σύμφωνα με τους Werbach και Hunter (2012) τα παιχνίδια έχουν ένα μοναδικό τρόπο να δημιουργούν και να διατηρούν το ενδιαφέρον των παικτών, είτε είναι αρχάριοι, είτε έχουν εξελιχθεί σε πιο επιδέξιους παίκτες. Στοιχεία των παιχνιδιών όπως η επίλυση προβλημάτων, η ανατροφοδότηση σχετικά με την πρόοδο μίας προσπάθειας, καθώς και η θετική ενθάρρυνση, φαίνεται πως πάντα έχουν πολύ θετική επίδραση στον ανθρώπινο εγκέφαλο, δημιουργώντας μία αίσθηση ευχαρίστησης και κατά συνέπεια δέσμευσης(Werbach & Hunter, 2012).

Η δύναμη των παιχνιδιών απορρέει από την επίδρασή τους σε τρεις τομείς: το γνωσιακό, το συναισθηματικό και τον κοινωνικό.

Στον γνωστικό τομέα, ο χρήστης εκπαιδεύεται, αποκτά γνώσεις πιο εύκολα από τι με την χρήση ενός βιβλίου καθώς χρησιμοποιεί τα οπτικοακουστικά μέσα.

Στον συναισθηματικό τομέα, ο χρήστης ενθαρρύνεται , εμψυχώνεται χωρίς αρνητική κρίση και διασκεδάσει. Από τα τέσσερα είδη διασκέδασης (εύκολη, δύσκολη, σοβαρή, ανθρώπινος παράγοντας) του Lazzaro (2004), εμείς θα ασχοληθούμε την εύκολη. Δηλαδή, ο παίκτης χαλαρώνει , εξερευνεί από περιέργεια και δεν είναι απαραίτητο να είναι πολύ δύσκολο έως να μην περιέχει καθόλου δυσκολίες. Μας ενδιαφέρει ο παίκτης εκτός του να μαθαίνει να διασκεδάσει ταυτόχρονα με το παιχνίδι, διότι αυτός είναι ο στόχος μάθηση μέσω ηλεκτρονικού παιχνιδιού. Για αυτό η παιχνιδοποίηση, αξιοποιεί στοιχεία των παιχνιδιών, όπως οι πόντοι, τα επίπεδα και οι πίνακες κατάταξης, σε περιβάλλοντα που δεν είναι παιχνίδια, με στόχο την ενίσχυση των κινήτρων και της συμμετοχής.

Στον κοινωνικό τομέα, η δυνατότητα ανάληψης διαφορετικών ρόλων, δίνει τη δυνατότητα στους παίκτες να υποδυθούν χαρακτήρες που δεν θα μπορούσαν ή θα δίσταζαν να δοκιμάσουν στην καθημερινότητά τους.

Σύμφωνα με το βιβλίο του Johan Huizinga (1938, “Homo Ludens”), διατυπώνει πως « το παιχνίδι είναι μία ελεύθερη δραστηριότητα που συνειδητά παραμένει ως κάτι «μη σοβαρό», εκτός της καθημερινότητας ενώ παράλληλα απορροφά τον παίχτη έντονα και ολοκληρωτικά. Είναι μια δραστηριότητα που έχει τα δικά του όρια χρόνου και χώρου και βασίζεται σε κανόνες. Προωθεί επίσης, τη δημιουργία κοινωνικών γκρουπ σε ένα περιβάλλον που λειτουργεί με διαφορετικό, δικό του τρόπο και μυστικότητα από τον υπόλοιπο κόσμο.» Το σχόλιο αυτό είναι επίκαιρο καθώς ο ρόλος των παιχνιδιών είναι έντονος διευρύνοντας το πεδίο εφαρμογής του από τα παιχνίδια των μικρών παιδιών έως τον εργασιακό τομέα.

Τα ηλεκτρονικά παιχνίδια έχουν αρκετές κατηγορίες όπως, παιχνίδια περιπέτειας (ακολουθούν μια ιστορία, περιλαμβάνουν προβλήματα για επίλυση, γρίφους), λογικής (στα οποία ο παίκτης πρέπει να αντιδρά πιο γρήγορα ώστε να πετύχει π.χ. μια μαθηματική εξίσωση), online (τα οποία ο παίκτης μπορεί να παίξει διαδικτυακά και με άλλους παίκτες) , serious games, παιχνίδια με βάση τοποθεσίας και city escape (παιχνίδια διαφυγής).

Οι τρεις κατηγορίες που θα αναλυθούν στην εργασία αυτή είναι τα serious,escape και urban games.

Θα χρησιμοποιηθούν ώστε ο παίκτης να κινείται καθώς μαθαίνει με μία ιστορία, σαν κίνητρο και να προσελκύσει το ενδιαφέρον του.

Κατηγορίες Ηλεκτρονικών Παιχνιδιών

Αρχικά, τα παιχνίδια **σοβαρού σκοπού (Serious games)**, τα οποία αποτελούν πρόδρομο της παιχνοποίησης, είναι ολοκληρωμένα παιχνίδια τα οποία χρησιμοποιούνται για λόγους που δεν αφορούν στη διασκέδαση. Στόχος τους είναι να παρέχουν πληροφορίες, να εκπαιδεύουν και να ευαισθητοποιηθούν για μια ιδέα ή άποψη πέρα από την διασκέδαση και την ψυχαγωγία.

Η χρήση τους έχει αυξηθεί, ιδίως σε τομείς όπως η εκπαίδευση, η άμυνα, η αεροναυτική, η επιστήμη ή η υγεία. Ο σκοπός τους μπορεί να είναι ένας από τους πολλούς: από την εκπαίδευση των πληρωμάτων πυροσβεστών σε καταστάσεις έκτακτης ανάγκης έως την εκπαίδευση μιας ομάδας πωλήσεων, τη διδασκαλία μαθηματικών ή την εξάσκηση μιας γλώσσας. Αντίστοιχα, για την εργασία μας να «εκπαιδευτούν» οι παίκτες, να μάθουν για την ιστορία ενός τόπου, ενός κτηρίου, την ιστορία του, πέραν από αυτό που βλέπουν και προσπερνούν καθημερινά.

Η μέθοδος που χρησιμοποιεί βιντεοπαιχνίδια για μαθησιακούς σκοπούς είναι γνωστή ως μάθηση με βάση το παιχνίδι. Οι υποστηρικτές αυτής της καινοτόμου μεθόδου διδασκαλίας πιστεύουν ότι τα βιντεοπαιχνίδια μπορούν να είναι ένα διασκεδαστικό και αποτελεσματικό εργαλείο ταυτόχρονα, μειώνοντας το κόστος των προγραμμάτων κατάρτισης, αυξάνοντας τα κίνητρα των μαθητών και διευκολύνοντας την άμεση πρακτική.

Δεν είναι εύκολο να δημιουργηθεί ένα τέτοιο παιχνίδι: απαιτεί πολύ ταλέντο, γνώσεις, χρόνο και πολλά χρήματα. Παρόλο που υπάρχει μια τεράστια ποικιλία μορφών και στόχων, τα περισσότερα συστήματα μάθησης με βάση το παιχνίδι τείνουν να έχουν πέντε στοιχεία που διασφαλίζουν ότι λειτουργούν σωστά και είναι αποτελεσματικά:

1. Μια ιστορία. Αν και δεν είναι καθόλου απαραίτητο, τα περισσότερα βιντεοπαιχνίδια έχουν μια πλοκή ή μια κύρια ιστορία.
2. Παιχνίδι. Είναι η δυναμική του παιχνιδιού, η οποία περιλαμβάνει τα συστήματα κατάταξης, ανταμοιβών, σημάτων ή πόντων. Αυτή η gamification τείνει να ζωντανεύει και να παρακινεί τους παίκτες με υγιή ανταγωνισμό.
3. Άμεση και εξατομικευμένη ανατροφοδότηση. Ο παίκτης αλληλοεπιδρά απευθείας με το παιχνίδι και λαμβάνει αμέσως ανταμοιβή ή τιμωρία. Στα πιο εξελιγμένα βιντεοπαιχνίδια, αυτά τα σχόλια είναι επίσης λεπτομερή και υποστηρίζονται: οι χρήστες μπορούν να γνωρίζουν πού έχουν πάει στραβά και μπορούν να προσπαθήσουν να κάνουν καλύτερα την επόμενη φορά. Εάν μεταδοθεί σωστά, αυτά τα σχόλια μπορούν να είναι ένα ισχυρό εργαλείο μάθησης.
4. Προσομοίωση. Στις περισσότερες περιπτώσεις, τα σοβαρά παιχνίδια αναπαράγουν ή μιμούνται πραγματικές καταστάσεις. Χρησιμοποιώντας πλασματικούς χαρακτήρες και αναδημιουργώντας σενάρια, οι παίκτες βρίσκονται βυθισμένοι σε έναν κόσμο παρόμοιο με αυτόν που θα βρουν πέρα από την οθόνη του υπολογιστή ή του κινητού τους. Αυτοί οι προσομοιωτές επιτρέπουν στους χρήστες να αλληλοεπιδρούν με μια νέα πραγματικότητα και να εξασκούν τις δεξιότητες και τις έννοιες που έχουν αποκτήσει κατά τη διάρκεια του παιχνιδιού.
5. Ο στόχος της μάθησης. Το βασικό στοιχείο των σοβαρών παιχνιδιών είναι ότι σκοπός τους είναι να διδάξουν κάτι. Εκτός από τη χρήση όλων αυτών των στοιχείων που αναφέρθηκαν, τα σοβαρά παιχνίδια πρέπει επίσης να έχουν έναν σκοπό που δεν είναι ψυχαγωγικό και σχεδόν πάντα να σχετίζεται με μια συγκεκριμένη πτυχή εκπαίδευσης ή κατάρτισης.

Όταν όλα τα παραπάνω στοιχεία συνδυάζονται επιτυχώς, έχουμε ένα ισχυρό εργαλείο εκμάθησης. Με την χρήση μιας αφήγησης, τεχνικών gamification, προσομοιωτών και ανατροφοδότησης κατά τη διαδικασία της διδασκαλίας συνεπάγεται πολλά οφέλη σε σύγκριση με τη μάθηση πρόσωπο με πρόσωπο ή την ηλεκτρονική μάθηση.

Τα πέντε πιο σημαντικά πλεονεκτήματα είναι :

Αυξάνουν την αφοσίωση και τα κίνητρα ενδιαφέροντος με την χρήση οπτικοακουστικών μέσων απομακρύνοντας ατέλειωτες μονότονες θεωρίες που άφηναν τον εκπαιδευόμενο αδιάφορο.

Επειδή ο κόσμος ζει σε μια μονότονη καθημερινότητα η χρήση ενός σοβαρού παιχνιδιού του δίνει την δυνατότητα να ξεφεύγει και να αισθάνεται μια ευχαρίστηση.

Διευκολύνουν την πραγματική και ασφαλή πρακτική. Χάρη στους προσομοιωτές, οι παίκτες μπορούν να ταυτιστούν με άλλους χαρακτήρες και μπορούν να χρησιμοποιήσουν τις γνώσεις που έχουν αποκτήσει από την αρχή. Τα σοβαρά παιχνίδια προσφέρουν επίσης τα καλύτερα και των δύο κόσμων: από τη μία πλευρά, οι προσομοιωτές θα πρέπει να αναπαράγουν την πραγματικότητα όσο πιο πιστά γίνεται από την άλλη πλευρά, το γεγονός ότι είναι ένα βιντεοπαιχνίδι επιτρέπει στους χρήστες να ασκούνται με ασφάλεια.

Βελτιώνουν την απομνημόνευση και τη διατήρηση, αυξάνοντας την αφοσίωση και τα κίνητρα των μαθητών, αυξάνουν επίσης την απομνημόνευσή τους και τη διατήρηση αυτών που έχουν μάθει.

Κάνουν τον κόσμο ένα καλύτερο μέρος, Αυτό μπορεί να ακούγεται υπερβολικά φιλόδοξο, αλλά το γεγονός είναι ότι πολλά σοβαρά παιχνίδια ήταν επίσης τόσο τολμηρά ώστε να προσπαθήσουν να αλλάξουν τον κόσμο. Το γεγονός ότι μας αρέσει το gamification και οι προσομοιωτές μπορεί να είναι ένα καλό εργαλείο για την ενθάρρυνση της εν συναίσθησης, την προώθηση της ειρήνης, την υπεράσπιση των δικαιωμάτων των ανθρώπων ή τη διάδοση αξιών του πολίτη. Σε αντίθεση με το στερεότυπο με το οποίο τα βιντεοπαιχνίδια αποτελούν αιτία αντικοινωνικής ή βίαιης συμπεριφοράς, η πραγματικότητα είναι ότι η μάθηση με βάση το παιχνίδι μπορεί να ενθαρρύνει την κοινωνική και θετική στάση για την κοινωνία στο σύνολό της.

Τα σοβαρά παιχνίδια είναι επίσης μια πολύ σοβαρή επιχείρηση. Σύμφωνα με εκτιμήσεις που έγιναν από την εταιρεία συμβούλων Marketandmarkets, ο τομέας αυτός θα αναπτυχθεί μεταξύ 2015 και 2020 με μέσο ετήσιο ρυθμό 16,38%, πράγμα που σημαίνει ότι η επιχείρηση θα έφτανε τα 5,449 δισεκατομμύρια δολάρια. Υπάρχει μια αυξανόμενη ζήτηση για μάθηση μέσω βιντεοπαιχνιδιών και σιγά-σιγά, η αγορά ανταποκρίνεται σε αυτή τη ζήτηση.

Η επόμενη κατηγορία που μας ενδιαφέρει είναι τα λεγόμενα **“αστικά” παιχνίδια (urban games)** ή αλλιώς **“παιχνίδια βάση τοποθεσίας”(location based)**.

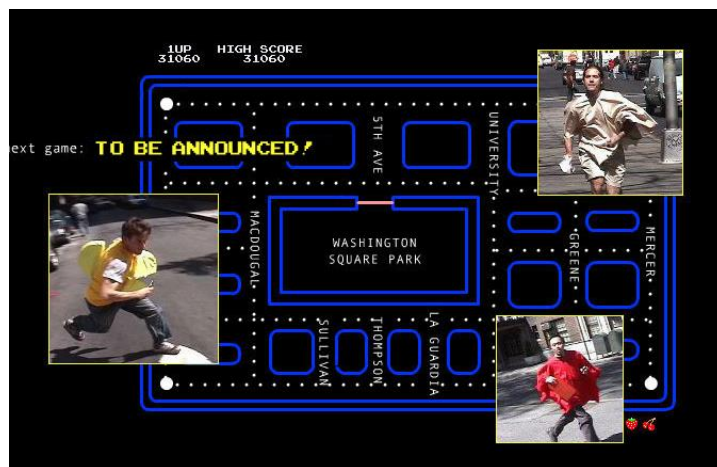
Τα κινητά τηλέφωνα και τα παγκόσμια συστήματα θέσης (GPS) χρησιμοποιούνται για να μπορεί ο χρήστης να λαμβάνει κλήσεις όταν βρίσκετε μακριά από το σπίτι και να μπορεί εύκολα να βρίσκει μέρη ή άτομα, συμπεριλαμβανομένου του εαυτού του σε περίπτωση που έχει χαθεί. Τα παιχνίδια αυτά παίζονται στον πραγματικό κόσμο χρησιμοποιώντας συσκευές χειρός, όπως κινητά τηλέφωνα καθώς τα καθιστά εύκολα στη χρήση τους σε σχέση με τους ηλεκτρονικούς υπολογιστές λόγω του βάρους τους. Λαμβάνοντας παιχνίδια σε αστικούς δρόμους, αυτά τα παιχνίδια χρησιμοποιούν τις τεχνολογίες SMS, ίντερνετ και GPS για να επιτρέψουν στους παίκτες να συμμετάσχουν στο αστικό τοπίο με διαφορετικό τρόπο.

Άνθρωποι σε όλο τον κόσμο χρησιμοποιούν αυτές τις τεχνολογίες για καινοτόμο χρήση, δημιουργώντας "εμπειρίες" σε δημόσιες ρυθμίσεις που είναι μέρος παιχνιδιού, μέρος τέχνης επιδόσεων και μέρος πειράματος κοινωνιολογίας. Με την ονομασία αστικών παιχνιδιών, αυτές οι δραστηριότητες μπορούν να κυμαίνονται από την αναδημιουργία του βιντεοπαιχνιδιού παραδείγματος χάριν το "Pac-Man" της δεκαετίας του '80 σε ανθρώπινη κλίμακα στη Νέα Υόρκη έως την απόκρυψη "θησαυρών" σε κρατικά πάρκα για να βρουν οι άνθρωποι μέσω GPS.

Οι στόχοι των διοργανωτών των παιχνιδιών με βάση την τοποθεσία δεν είναι μόνο οι διασκέδαση αλλά η κοινωνικοποίηση και η μάθηση χρήσης νέων τεχνολογιών αλλά και μάθηση τοπίων, έργων τέχνης κλπ.

Η δημιουργία πολλών παιχνιδιών προήλθε από επιτραπέζια παιχνίδια και εφαρμόστηκαν σε αστικά τοπία.

Η πλειοψηφία των αστικών παιχνιδιών πραγματοποιείται στην Ευρώπη. Ωστόσο, ένα δημοφιλές παιχνίδι παίζεται γύρω από το Washington Square Park της Νέας Υόρκης. Στο "Pac-Manhattan". Ένα παιχνίδι που βασίζεται στο κινητό και οι παίκτες φορούν κοστούμια που αντιπροσωπεύουν χαρακτήρες από το Pac-Man, το κλασικό παιχνίδι arcade. Ο παίκτης Pac-Man πρέπει να τρέχει γύρω από τους δρόμους της πόλης, να «στροβιλίζει» εικονικές κουκίδες αποφεύγοντας τα φαντάσματα που προσπαθούν να τον φάνε.



<http://www.pacmanhattan.com/>

Η αιτία που ξεκίνησαν τέτοιου είδους παιχνίδια είναι επειδή οι χώροι όπου τα παιδιά των αστικών κέντρων μπορούν να παίξουν ελεύθερα μειώνονται συνεχώς σε πολλές χώρες του κόσμου. Καθώς η ταχεία αστικοποίηση τροποποιεί το φυσικό και το κοινωνικό περιβάλλον των πόλεων, τα παιδιά σήμερα δεν έχουν στη διάθεσή τους ούτε τον φυσικό ούτε τον ψυχολογικό χώρο για πραγματικά ελεύθερο παιχνίδι (Goodenough 2003, Atmakur-Javdekar 2016). Ο φόβος για την ασφάλεια των παιδιών, η υπερπροστατευτικότητα των γονέων και το υπερφορτωμένο πρόγραμμα 'εξωσχολικών' δραστηριοτήτων παιδιών και γονέων οδηγούν τις οικογένειες να αναζητούν κοντινούς, 'φροντισμένους' χώρους παιχνιδιού, με σαφή όρια και κανόνες ασφάλειας,

Έτσι και στην Ελλάδα, μία ομάδα ίδρυσε το «Pop-Up Adventure Playgrounds». Η διάρκεια της λειτουργίας των συγκεκριμένων χώρων ποικίλει ανάλογα με τους στόχους ή/και τις δυνατότητες του φορέα που οργανώνει τη δράση (π.χ. δήμος, σχολεία, ιδιωτική πρωτοβουλία) καθώς και τα χαρακτηριστικά του χώρου που τη φιλοξενεί (π.χ. εσωτερικός ή υπαίθριος, δημόσιος ή ιδιωτικός χώρος). Τα υλικά που χρησιμοποιούνται στα pop-up adventure playgrounds είναι καθημερινά, ανοιχτά, ευέλικτα, φυσικά ή συνθετικά υλικά τα οποία μπορούν να μετακινηθούν, να συνδυαστούν, να διαλυθούν και να 'ξενασχεδιαστούν' με πολλαπλούς τρόπους. Χαρακτηριστικά παραδείγματα ανοιχτών υλικών αποτελούν τα λάστιχα αυτοκινήτων, τα χάρτινα κιβώτια, οι καρποί δέντρων (π.χ. κουκουνάρια), τα κοχύλια, το νερό, η άμμος, οι πλαστικοί σωλήνες, δοχεία διαφόρων ειδών, κομμάτια ύφασμα, ανακυκλώσιμα αντικείμενα και συσκευές κ.ά.

Η διεθνής έρευνα και οι εμπειρίες στα τοπικά μέσα δείχνουν πώς τα αστικά παιχνίδια μπορούν να είναι κρίσιμο στοιχείο στη σχέση μεταξύ πόλεων και ανθρώπων. μια εμπειρία παιχνιδιού θα μπορούσε να τροποποιήσει τη σύνδεση μεταξύ της πόλης και των κατοίκων της, καθιστώντας την πιο πλούσια, πιο ενδιαφέρουσα (Μπόρις 2007) . Τα αστικά παιχνίδια έχουν αποδείξει ότι έχουν τη δύναμη να αλλάξουν

την αντίληψή μας για τους χώρους ρουτίνας και, ταυτόχρονα, να προσθέσουν νέες έννοιες στην ανθρώπινη ερμηνεία του φυσικού περιβάλλοντος. Ωστόσο, τα πιθανά οφέλη του αστικού παιχνιδιού είναι πολλά. Η κοινωνική αλληλεπίδραση και τα συναισθήματα της περιπέτειας κατά την εξερεύνηση νέων ρυθμίσεων ή την προσπάθεια επίλυσης παζλ μπορεί να είναι πολύ ικανοποιητική.

Ορισμένα αστικά παιχνίδια εκτελούνται σαφώς εντός ενός σημείου, καθώς μπορεί να είναι δύσκολο για έναν ανθρώπινο παίκτη να εντοπίσει τα όρια σε παιχνίδια που διαμορφώνονται χωρικά σε τόσο μεγάλες περιοχές που το μάτι δεν μπορεί να τα αντιληφθεί στο σύνολό τους.

Προβλήματα χρήσης

Καθώς οι συσκευές με δυνατότητα GPS καθίστανται συνηθισμένες, τα αστικά παιχνίδια αναμένεται να αυξηθούν σε δημοτικότητα. Ήδη οι εταιρείες επικοινωνιών χρησιμοποιούν αστικά παιχνίδια (ή παιχνίδια βάσει τοποθεσίας), για να προσελκύσουν τους καταναλωτές σε συνδρομές ή ως καμπάνιες μάρκετινγκ για να προσελκύσουν την αγορά των νέων. Ωστόσο, όλη αυτή η καινοτομία στα παιχνίδια δεν προχωρά ανεμπόδιστη. Τα συστήματα GPS χρειάζονται ένα οπτικό πεδίο για να λειτουργήσει ένας δορυφόρος. Αυτό σημαίνει ότι δεν λειτουργούν σε εσωτερικούς χώρους ή σε αστικά περιβάλλοντα που περιβάλλονται από ψηλά κτίρια.

Η τεχνολογία οδηγεί σε ένα άλλο πρόβλημα: αυτό το είδος παιχνιδιού μπορεί να είναι πολύ ακριβό. Ροές ήχου ή βίντεο σε πραγματικό χρόνο, συνδέσεις στο Διαδίκτυο για κινητά και λεπτά κινητού τηλεφώνου μπορεί να κοστίσουν ένα πακέτο, ακόμη και σε ένα παιχνίδι που διαρκεί μόνο μία ώρα. Τα δίκτυα δεν είναι πάντα τέλεια. Μια αργή ή μειωμένη σύνδεση μπορεί να σταματήσει ολόκληρο το παιχνίδι. Άλλα προβλήματα περιλαμβάνουν το παιχνίδι σε δημόσιους χώρους. Θα υπάρχουν άνθρωποι γύρω που δεν γνωρίζουν ότι υπάρχει ένα παιχνίδι γύρω τους, και μπορεί να μπερδευτούν ή να θυμώσουν με τους παίκτες. Είναι επίσης δυνατό για τα παιχνίδια να οδηγήσουν τους παίκτες σε μη ασφαλείς περιοχές. Μερικοί από αυτούς τους παράγοντες μπορούν να παρακολουθούνται από "διαιτητές" που επιβλέπουν το παιχνίδι.

Επίσης, επειδή δεν υπάρχει απόλυτος έλεγχος στη σχεδίαση επιπέδων, επειδή εξαρτάται από την τοποθεσία του παίκτη, είναι πολύ δύσκολο για τους προγραμματιστές να εγγραφούν μια ομοιόμορφη εμπειρία παιχνιδιού. Επιπλέον, η φυσική ικανότητα του παίκτη πρέπει να ληφθεί υπόψη εάν το παιχνίδι βάσει τοποθεσίας απαιτεί από τον παίκτη να μετακινηθεί στον πραγματικό κόσμο.

Σε περίπτωση που η τοποθεσία του παίκτη είναι προσδιορίσιμη, εάν υπάρχουν δεδομένα σχετικά με την τοποθεσία για πρόσβαση, όπως χάρτες, καιρός ή άλλες πληροφορίες, μερικές φορές δεν είναι διαθέσιμο για την περιοχή του παίκτη ή δεν περιέχει αρκετά δεδομένα σχετικά με το παιχνίδι. Επίσης, η πρόσβαση σε αυτές τις πληροφορίες σημαίνει ότι είτε αποθηκεύονται τοπικά, πράγμα που σημαίνει ότι θα είναι τελικά ξεπερασμένο ή περιορισμένο μόνο σε ορισμένες τοποθεσίες, καθώς η χωρητικότητα αποθήκευσης αυτών των κινητών συσκευών.

Τέλος, ένα ζήτημα που αποτρέπει ορισμένα άτομα είναι η χρήση των δεδομένων που σχετίζονται με την τοποθεσία του προγράμματος αναπαραγωγής (player's location-related data). Όπως είναι γνωστό, τα ζητήματα απορρήτου έχουν σημασία, ειδικά όταν σχετίζονται με παιχνίδια βάσει τοποθεσίας, καθώς αυτά τα παιχνίδια είναι πιθανό να παίζονται από παιδιά. Οι εφαρμογές που επιτρέπουν την κοινή χρήση των προσωπικών πληροφοριών του χρήστη αναμένεται να χειριστούν αυτές τις πληροφορίες με προσοχή και να ενημερώνουν ρητά τον χρήστη πώς χρησιμοποιούνται και για ποιο σκοπό.

Λύσεις

Τα προαναφερθέντα ζητήματα μπορούν να συνοψιστούν ως εξής: ζητήματα σχεδιασμού παιχνιδιού, περιορισμοί υλικού, διαθεσιμότητα και καταλληλότητα πληροφοριών σχετικά με την τοποθεσία και προστασία δεδομένων του παίκτη. Υπάρχουν όμως αρκετές λύσεις για αυτά τα προβλήματα.

Όντας ένα παιχνίδι βάσει τοποθεσίας, συνεπάγεται πάντα κάποιο βαθμό απρόβλεπτης συμπεριφοράς από τον παίκτη. Αυτό ελαττώνεται καθορίζοντας συγκεκριμένες περιοχές που μπορεί να παίζει ο χρήστης.

Όσο αφορά τα προσωπικά δεδομένα του χρήστη, Όταν παίζετε τα περισσότερα παιχνίδια βάσει τοποθεσίας, οι πληροφορίες τοποθεσίας του χρήστη, είτε είναι συντεταγμένες είτε πλήρως αναγνωσμένες διευθύνσεις, συχνά αποθηκεύονται προσωρινά στην αποθήκευση είτε για μελλοντική χρήση είτε για στατιστικούς σκοπούς. Έτσι

Η προτεινόμενη προσέγγιση είναι η αποθήκευση αυτών των πληροφοριών είτε εξ αποστάσεως, δηλαδή θα πρέπει να ζητηθεί από τον χρήστη να εισαγάγει ένα όνομα χρήστη και έναν κωδικό πρόσβασης πριν από το παιχνίδι. Επίσης, να υπάρχει ειδοποίηση για ανίχνευση της τοποθεσίας του παίκτη για να μπορεί να παίζει το παιχνίδι.

Σκοπός του παιχνιδιού είναι η ιστορική εξερεύνηση και μάθηση του αστικού ιστού της Κέρκυρας. Μέσα από την εξερεύνηση αυτή ο παίκτης θα μάθει , θα ενημερωθεί για τα ιστορικά αξιοθέατα της πόλης όπου μέχρι σήμερα μπορεί να πέρναγε από μπροστά τους αγνοώντας τα.

Η δημιουργία του παιχνιδιού στηρίζεται σε έναν χάρτη της πόλης της Κέρκυρας, όπου έχει προσημειωμένα τα ιστορικά μνημεία σε μορφή πληροφοριών και ερωτήσεων γνώσεις με βάση τις πληροφορίες αυτές. Ο παίκτης μπορεί ελεύθερα να ακολουθήσει την επιθυμητή διαδρομή με συγκεκριμένο τέλος. Επίσης , υπάρχουν σημεία που λειτουργούν σαν μπόνους στην βαθμολογία (το νόμισμα) αλλά ταυτόχρονα και σημεία που αφαιρούν πόντους (η νεκροκεφαλή). Υπάρχει ένα σημείο στο χάρτη (σπαθιά) που εάν ο παίκτης δεν έχει μαζέψει ένα σκορ 30 πόντων εμφανίζεται ένα μήνυμα ότι αιχμαλωτίστηκε ο παίκτης από τους πειρατές. Εάν η βαθμολογία είναι μεγαλύτερη των 30 πόντων τότε εμφανίζεται ένα “κρυφό” κουίζ ερωτήσεων για επιπλέον πόντους. Στόχος είναι η συγκέντρωση της μεγαλύτερης βαθμολογίας για την ολοκλήρωση του παιχνιδιού. Ο τελικός έλεγχος γίνεται στο σημείο που έχει εικονίδιο ένα σεντούκι.

Το παραπάνω σενάριο χρήσης της εφαρμογής δεν είχε πρότυπο άλλου παιχνιδιού για να ακολουθήσει για αυτό και δημιουργήθηκε εξ ολοκλήρου από την αρχή.

Η εργασία αυτή απευθύνεται σε όλες τις ηλικίες του πληθυσμού , μαθητές , εφήβους, τουρίστες και ντόπιους κατοίκους, εκτός νηπιακής ηλικίας. Θα μπορούσε βέβαια σύμφωνα με την ηλικία του χρήστη παραδείγματος χάριν παιδική ηλικία, αντί για ιστορικά μνημεία που πιθανόν να μην υπάρχει ενδιαφέρον από το παιδί ο χάρτης της εφαρμογής να έχει επιλογή για να βρει παιδότοπους, παιδική χαρά κλπ.

Οι απαιτήσεις που τέθηκαν λειτουργικές είναι το αρχικό μενού, ο χάρτης που εμφανίζεται, τα σημεία των ιστορικών μνημείων, οι ερωτήσεις, οι απαντήσεις και η βαθμολογία.

Οι απαιτήσεις που δεν τέθηκαν σε λειτουργία είναι : δεν έχει εφαρμογή σε κινητή συσκευή, δεν υπάρχει ένα σύστημα σύνδεσης των παικτών και τα «πατημένα» σημεία δεν έχουν χρωματική αλλαγή ώστε να καταλαβαίνει ο παίκτης ότι έχει δει αυτές τις πληροφορίες/ερωτήσεις.

Το παιχνίδι εάν ολοκληρωθεί και είναι λειτουργικό θα ήταν επιθυμητό να παίζεται κυρίως σε φορητή συσκευή γιατί ο σκοπός του είναι η εξερεύνηση των ιστορικών μνημείων και σημείων της πόλης για να αποκτήσουν οι χρήστες την γνώση αυτών και ταυτόχρονα να κινούνται μέσα στην πόλη.

Το επιδιωκόμενο επιθυμητό αποτέλεσμα είναι οι χρήστες να μάθουν αρχικά να κινούνται εντός της πόλης με γνώμονα τα ιστορικά μνημεία , να αφομοιώσουν τις γνώσεις αυτές ευχάριστα με απλό τρόπο χωρίς κάποιο αντίτιμο πέραν της γνώσης. Βέβαια εφικτό θα ήταν το παιχνίδι να έχει χορηγό παραδείγματος χάριν ένα κατάστημα καφέ όπου σαν ανταμοιβή θα παρέχει μια έκπτωση εφόσον έχουν συμπληρωθεί κάποια μπόνους μέχρι να φτάσει στο κατάστημα αυτό.

Επιπλέον, θα μπορούσε ο κάθε παίκτης να συνδέεται με ένα όνομα χρήστη και κωδικό για να μπορεί να παίζει το παιχνίδι αλλά και να επικοινωνεί με άλλους παίκτες αφήνοντας τους μηνύματα σε διάφορα σημεία της πόλης και κάθε μήνα ο χρήστης με το υψηλότερο σκορ να κερδίζει δύο εισιτήρια σε ένα από τα ιστορικά μνημεία που αναγράφονται στο χάρτη του παιχνιδιού.

Αξιολόγηση

Δημιουργήθηκαν δύο ερωτηματολόγια για την εργασία αυτή. Ένα στην αρχή πριν δημιουργηθεί το παιχνίδι για ιδέες και ένα στο τέλος για εντυπώσεις και συμβουλές για πιθανές μελλοντικές βελτιώσεις.

Και τα δύο ερωτηματολόγια μοιράστηκαν σε πάνω από δέκα ανθρώπους και διεξήχθησαν ηλεκτρονικά μέσω της πλατφόρμας της Google, το “Google φόρμες”. Στην επικεφαλίδα του καθενός ερωτηματολογίου, υπάρχει μία μικρή ενημέρωση γραπτός για το θέμα του παιχνιδιού και για ποιο λόγο γίνονται αυτές οι ερωτήσεις. Η συμμετοχή των εθελοντών που ερωτώνται είναι ανώνυμη. Επίσης, οι παίκτες ήταν ηλικιών από 15 έως 50 ετών και τους κοινοποιήθηκε μία διεύθυνση url, από την πλατφόρμα “GoormIDE”, για να μπορούν να δουν την εφαρμογή διαδικτυακά χωρίς να χρειάζεται να εγκαταστήσουν ειδικά εργαλεία (π.χ. Node.js τερματικό).

Οι ερωτήσεις του πρώτου ερωτηματολογίου,

1. Να εμφανίζονται ερωτήσεις που θα σας καθοδηγούν πως θα προχωρήσετε στο χάρτη;
☐ Ναι
☐ Όχι
2. Να επιλέγεται εσείς πως θα θέλατε να κινηθείτε στο χάρτη (Να επιλέγετε σε ποιο αστέρι θέλετε να πάτε)
☐ Ναι
☐ Όχι
3. Υποχρεωτικά να πηγαίνετε από το ένα αστέρι στο άλλο
☐ Ναι
☐ Όχι
4. Να είναι παιχνίδι μόνο με ερωτήσεις και κουίζ (Ναι/Όχι και γιατί)

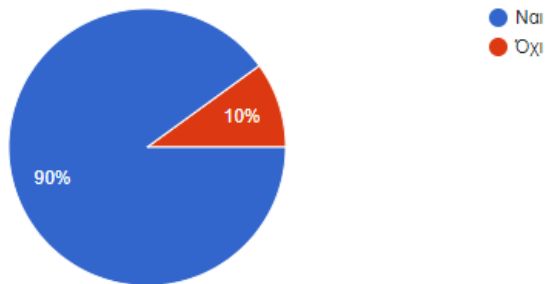
Κείμενο σύντομης απάντησης

5. Δικές σας ιδέες

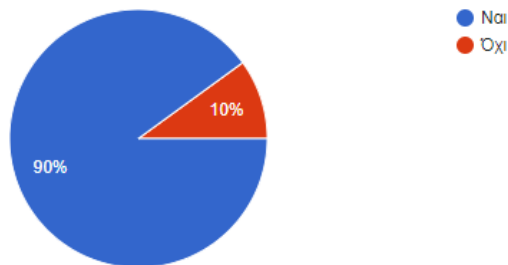
Κείμενο μακροσκελούς απάντησης

Οι απαντήσεις ,

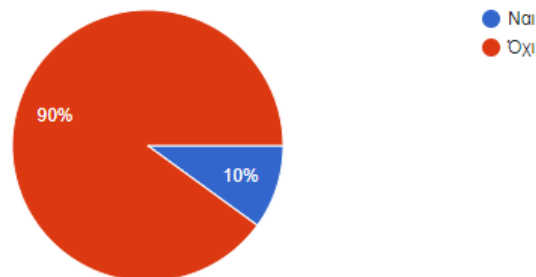
1. Να εμφανίζονται ερωτήσεις που θα σας καθοδηγούν πως θα προχωρήσετε στο χάρτη;



2. Να επιλέγεται εσείς πως θα θέλατε να κινηθείτε στο χάρτη (Να επιλέγετε σε ποιο αστέρι θέλετε να πάτε)



3. Υποχρεωτικά να πηγαίνετε από το ένα αστέρι στο άλλο



4. Να είναι παιχνίδι μόνο με ερωτήσεις και κουίζ (Ναι/Όχι και γιατί)

- ΟΧΙ
- Όχι γιατί θα είναι σχετικά απλό και θα μπορεί να το τερματίσει κάποιος γρήγορα.
- Όχι, τα κουίζ και οι ερωτήσεις είναι βαρετές
- Ναι, γιατί χρειάζονται εκπαιδευτικές ερωτήσεις σχετικά με τα περιεχόμενα των μουσείων κ όχι περιττά πράγματα
- Όχι απαραίτητα αν γινόταν να υπάρχει και κάποιο challenge τύπου αφού πάω στο σημείο να πρέπει να κάνω κάτι

- όχι γιατί τα κουίζ κουράζουν
- Ναι. Πιστεύω οι καλές ερωτήσεις βοηθάνε στο να θυμάται ο παίκτης πιο εύκολα συγκεκριμένες πληροφορίες.
- Όχι, γιατί έχει άπειρες δυνατότητες ένα παιχνίδι. Σκέψου διάφορα!
- Ναι γιατί είναι πιο ευχάριστο

5. Δικές σας ιδέες

- Σε κάθε μουσείο να περνάει μια δοκιμασία, πχ να πρέπει να βρει ένα κλειδί.
- Θα μπορούσες να βάλεις achievements(π.χ περπατά 10, 50, 100km εάν χρησιμοποιεί gps το παιχνίδι, επισκέψου όλα τα σημεία, απάντησε σωστά σε όλες τις ερωτήσεις του κουίζ κτλ...) ή και ένα πίνακα κατάταξης για να βλέπουν οι χρήστες τα σκορ των άλλων παιχτών για να δημιουργήσεις περισσότερο ανταγωνισμό.
- The secret ingredient is crime
- έξτρα περιεχόμενο μετά την ολοκλήρωση μιας διαδρομής από αστέρι σε αστέρι..να πετάγεται ειδοποίηση ότι κέρδισες βραβείο και πήρες πόντους μετά από την διαδρομή
- Ίσως η προβολή κάποιας πληροφορίας ενός εκθέματος να γίνεται και με τη συνοδεία ήχου έτσι ώστε ο χρήστης να μην είναι υποχρεωμένος να κοιτάει την οθόνη συνέχεια.
- Θα μπορούσε να έχει διάφορα στοιχεία και λεπτομέρειες κρυμμένες στην κάθε τοποθεσία/αστέρι. Δηλαδή πχ συνολικά αναζητάς 7 λεπτομέρειες/στοιχεία για να μπορέσεις να τελειώσεις το παιχνίδι και να κερδίσεις. Οι 7 αυτές πληροφορίες/λεπτομέρειες θα είναι κρυμμένες σε 7 αστέρια. Θα πρέπει να υπάρχουν και τοποθεσίες που δεν θα έχουν στοιχεία και θα είναι παραπλανητικές, ώστε να σε κάνουν να χάσεις χρόνο. Επίσης, μπορείς να έχεις κάποια δώρα μπόνους σε κάποια αστέρια (ανάλογα το μνημείο θα σου δίνει βραβείο πχ ένα άγαλμα, ένα χαρακτηριστικό μουσικό κομμάτι από την Κέρκυρα, ή μια φωτογραφία μιας όμορφης τοποθεσίας από την περιοχή του αστεριού). Δηλαδή σε μία τοποθεσία θα κερδίζεις πιο γρήγορη κίνηση ή κάποιο tip για να σε βοηθήσει σε κάποια άλλη τοποθεσία ή να είναι τοποθεσία παγίδα (να πηγαίνεις πιο αργά άμα πέσεις σε αυτήν). Επίσης μια τοποθεσία μπορεί να έχει ως tip ένα κανόνι που θα σε καθοδηγεί ότι πρέπει να πας στο κανόνι της Κέρκυρας. Ή μια τοποθεσία να σου δείχνει ένα φρούριο και να σε καθοδηγεί να πας στο παλιό ή στο νέο φρούριο από τα οποία το ένα θα έχει βραβείο ένα στοιχείο και το άλλο θα είναι χωρίς στοιχεία. Τέλος, μπορείς να βάλεις μια τοποθεσία Α η οποία θα λέει λάθος στοιχεία και σε μια άλλη τοποθεσία Β θα λέει ότι στο μέρος Α λέει ψέματα. Οι χαρακτήρες/πιόνια του παιχνιδιού θα μπορούσε να είναι τα εξής: ένας πειρατής, ένα φρούτο κουμκουάτ, ένα φρούριο μικρογραφία, μια κανάτα μικρή που ρίχνουν το Πάσχα, ένα μουσικό όργανο φιλαρμονικής (θα γράφεις μια μικρή ιστορία κάτω από κάθε επιλογή - πχ το κανάτι γιατί το ρίχνουν το Πάσχα στις γιορτές).

Ύστερα από τις απαντήσεις των χρηστών , συμπεράναμε ότι επιθυμούν να υπάρχει μία ελευθερία κίνησης στο παιχνίδι, δηλαδή να μην υποχρεώνονται να κινούνται με μία προκαθορισμένη σειρά. Το 50% απάντησε πώς τα κουίζ είναι βαρετά και πώς θα ήθελαν να έχει και κάποιο επιπλέον παιχνίδι, π.χ. τρίλιζα ενώ το άλλο 50% δήλωσε πώς ενδιαφέρονται μόνο για απλές ερωτήσεις γνώσεων. Τέλος, στην ερώτηση για προσωπικές επιθυμίες υπήρξαν αρκετές απόψεις οι οποίες μπορούν να ληφθούν υπόψιν στο μέλλον εάν η εφαρμογή ολοκληρωθεί σε κινητές συσκευές.

Εργαλεία Υλοποίησης

Τα τεχνολογικά εργαλεία/βιβλιοθήκες/γλώσσες προγραμματισμού που χρειάστηκαν και συντέλεσαν στην ολοκλήρωση του κώδικα παρουσιάζονται παρακάτω με επεξήγηση στο καθένα.

- | | |
|---------------------|------------------|
| ✓ HTML | ✓ React confetti |
| ✓ Javascript | ✓ Node.js |
| ✓ CSS | ✓ Npm |
| ✓ ReactDOM | ✓ Npm start |
| ✓ React ES6 | ✓ Json |
| ✓ React Components | ✓ React |
| ✓ React Google Maps | ✓ VSCode |
| ✓ React Bootstrap | ✓ Json |

Επίσης να αναφέρουμε πως τα παρακάτω εργαλεία επιλέχθηκαν να χρησιμοποιηθούν για την εφαρμογή μας ύστερα από έρευνα ότι ήταν τα πιο κατάλληλα και εφαρμόσιμα για την εργασία μας.



HTML

HTML Logo - Internet Images

HTML είναι το ακρόνυμο από το **H**yper **T**ext **M**arkup **L**anguage που σημαίνει γλώσσα χαρακτηρισμού υπερκειμένου. Η χρήση μιας γλώσσας χαρακτηρισμού σημαίνει ότι γράφεται πρώτα το κείμενο και έπειτα προστίθενται ειδικά σύμβολα γύρω από τις λέξεις ή από ολόκληρες προτάσεις ώστε να καθοριστεί η εμφάνιση τους στην οθόνη. Τα ειδικά σύμβολα στην HTML λέγονται ετικέτες (tags). Οι ετικέτες ελέγχουν την δομή και την μορφή του κειμένου της ιστοσελίδας. Επίσης παρέχουν πληροφορίες προς τον web browser για την σελίδα που πρόκειται να εμφανίσουν, όπως ο τίτλος της σελίδας κ.α.

Οι HTML ετικέτες γράφονται ανάμεσα στα σύμβολα < και >, <όνομα-ετικέτας>.

Παραδείγματος χάριν, στον κώδικα χρησιμοποιούνται τα <div>, όπου ορίζουν μια διαίρεση ή μια ενότητα. Χρησιμοποιήθηκαν ως ενότητα για στοιχεία HTML - τα οποία στη συνέχεια διαμορφώθηκαν με

CSS ή χειρίστηκαν με JavaScript. Επίσης, σε κάθε ετικέτα έγινε χρήση του χαρακτηριστικού className ώστε να διαμορφωθεί εύκολα.



JavaScript

Nick Coughlin JavaScript Notes - DOM Manipulation

Η JavaScript είναι μία γλώσσα προγραμματισμού υπολογιστών, για σεναριοποίηση των διαδραστικών αποτελεσμάτων εντός των περιηγητών ιστού. Αποτελεί μία από τις πιο ευέλικτες και αποτελεσματικές γλώσσες που μπορεί να χρησιμοποιηθούν από προγραμματιστές, όχι μόνο σε κάθε ιστότοπο, αλλά επίσης και σε ιστότοπους κινητών, παιχνιδιών και εφαρμογών ιστού.

Έγινε η χρήση αυτής της γλώσσας διότι γίνεται η επεξεργασία από την πλευρά του πελάτη. Δηλαδή, ο κώδικας εκτελείται από τον επεξεργαστή του χρήστη, αντί του εξυπηρετητή ιστού, εξοικονομώντας έτσι εύρος ζώνης και περιορίζοντας την υπερφόρτωση του εξυπηρετητή. Επίσης, δεν απαιτεί κανέναν ειδικό μεταγλωττιστή ή συντάκτη και τα αποσπάσματα κώδικα χρησιμοποιούνται στις ιστοσελίδες, όπου είναι απαραίτητο. Εύκολη είναι η προσθήκη της προσωπικότητας του προγραμματιστή και οι χρήστες του ιστότοπου δεν χρειάζονται ειδικό λογισμικό και λήψεις προγραμμάτων για να δουν τη JavaScript, έτσι κάθε χρήστης έχει την ίδια εμπειρία.



CSS

PNGEgg Css png images

Η CSS, Cascading Style Sheets είναι μια γλώσσα που περιγράφει τη μορφοποίηση ενός εγγράφου HTML. Η μορφοποίηση περιλαμβάνει τη σχεδίαση, την παρουσίαση καθώς και τις διαφοροποιήσεις εμφάνισης της ιστοσελίδας ανάλογα με τη συσκευή ή με το μέγεθος της οθόνης.

Η χρησιμοποίηση του CSS μας βοηθάει να γλιτώσουμε πολύ κόπο. Μπορούμε με ένα μόνο αρχείο να καθορίσουμε την εμφάνιση των στοιχείων της html, του οποίου η κατάληξή είναι .css.



React Js DOM

React official Logo from official React site

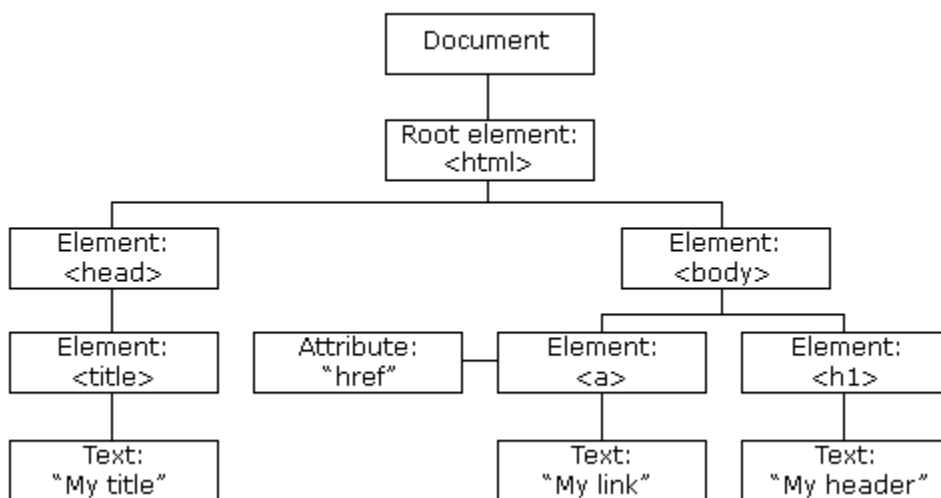
Το Document Object Model (DOM) είναι μια διεπαφή προγραμματισμού για έγγραφα HTML και XML. Αντιπροσωπεύει τη σελίδα έτσι ώστε τα προγράμματα να μπορούν να αλλάξουν τη δομή, το στυλ και το περιεχόμενο του εγγράφου. Το DOM αντιπροσωπεύει το έγγραφο ως κόμβους και αντικείμενα. Με αυτόν τον τρόπο, οι γλώσσες προγραμματισμού μπορούν να συνδεθούν στη σελίδα. Το DOM είναι μια αντικειμενοστραφή αναπαράσταση της ιστοσελίδας, η οποία μπορεί να τροποποιηθεί με μια γλώσσα δέσμης ενεργειών όπως τη JavaScript.

Το DOM είναι ένα πρότυπο W3C (World Wide Web Consortium) που ορίζει ένα πρότυπο για την πρόσβαση σε έγγραφα : "Το W3C Document Object Model (DOM) είναι μια πλατφόρμα και ουδέτερη γλωσσική διεπαφή που επιτρέπει σε προγράμματα και σενάρια να έχουν δυναμική πρόσβαση και ενημέρωση του περιεχομένου, της δομής και του στυλ ενός εγγράφου."

Το πρότυπο W3C DOM χωρίζεται σε 3 διαφορετικά μέρη:

- ❖ **Core DOM - πρότυπο μοντέλο για όλους τους τύπους εγγράφων**
- ❖ **XML DOM - τυπικό μοντέλο για έγγραφα XML**
- ❖ **HTML DOM - τυπικό μοντέλο για έγγραφα HTML**

Το HTML DOM (Μοντέλο αντικειμένου εγγράφου) όταν φορτώνεται μια ιστοσελίδα, το πρόγραμμα περιήγησης δημιουργεί ένα μοντέλο αντικειμένου εγγράφου της σελίδας και κατασκευάζεται ως δέντρο αντικειμένων:



JavaScript HTML DOM w3schools.com

Όταν θέλουμε να αποκτήσουμε πρόσβαση σε στοιχεία HTML με JavaScript, πρέπει πρώτα να βρούμε τα στοιχεία. Υπάρχουν κάποιοι τρόποι για να το κάνουμε αυτό:

- ⇒ **Εύρεση στοιχείων HTML ανά αναγνωριστικό (elements by id)**
- ⇒ **Εύρεση στοιχείων HTML ανά όνομα ετικέτας (elements by tag name)**
- ⇒ **Εύρεση στοιχείων HTML ανά όνομα τάξης (elements by class name)**
- ⇒ **Εύρεση στοιχείων HTML με συλλογές αντικειμένων HTML (HTML object collections)**
- ⇒ **Εύρεση στοιχείων HTML από επιλεγμένους CSS (CSS selectors)**

Ο ευκολότερος τρόπος για να βρεθεί ένα στοιχείο HTML στο DOM, είναι χρησιμοποιώντας το στοιχείο id. Παραδείγματος χάριν στον κώδικα μας, η μέθοδος Document.getElementById () επιστρέφει ένα

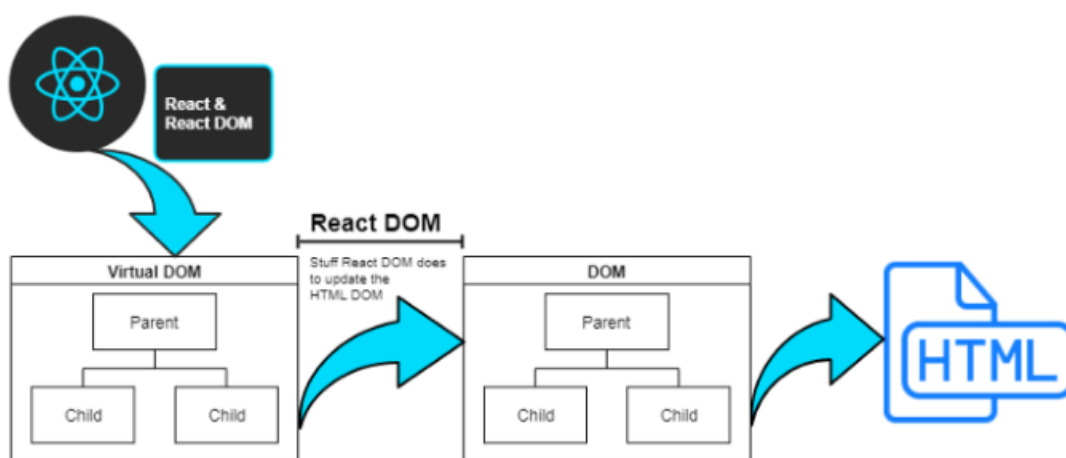
στοιχείο Element που αντιπροσωπεύει το στοιχείο του οποίου η ιδιότητα id ταιριάζει με την καθορισμένη συμβολοσειρά. Δεδομένου ότι τα αναγνωριστικά στοιχείων πρέπει να είναι μοναδικά εάν καθορίζονται, είναι ένας χρήσιμος τρόπος για γρήγορη πρόσβαση σε ένα συγκεκριμένο στοιχείο.

```
ReactDOM.render(<App />, document.getElementById("root"));
```

Για αυτό εδώ μπαίνει και το ReactDOM. Το React εφαρμόζει ένα σύστημα DOM ανεξάρτητο από πρόγραμμα περιήγησης για απόδοση και συμβατότητα μεταξύ προγραμμάτων περιήγησης παρέχοντας πάντοτε ένα API που εστιάζει στη JavaScript στο DOM.

Τα στοιχεία είναι τα μικρότερα δομικά στοιχεία των εφαρμογών React. Ένα στοιχείο περιγράφει τι θέλουμε να δούμε στην οθόνη.

Το React DOM φροντίζει να ενημερώσει το DOM ώστε να ταιριάζει με τα στοιχεία React.



React / ReactDOM — ELI5 / <https://medium.com/Gabriel Rasdale>

Για αυτό ονομάζουμε "ριζικό" κόμβο DOM ("root"), διότι τα πάντα στο εσωτερικό του θα διαχειρίζονται το ReactDOM.

```
ReactDOM.render(<Corfu />, document.getElementById("root"));
```

Οι εφαρμογές που έχουν δημιουργηθεί με το React έχουν συνήθως έναν μόνο κόμβο DOM ρίζας. Για να αποδώσουμε ένα στοιχείο React σε έναν ριζικό κόμβο DOM, περνάμε και τα δύο στο ReactDOM.render().

Το React DOM συγκρίνει το στοιχείο και τα παιδιά του με το προηγούμενο και εφαρμόζει μόνο τις απαραίτητες ενημερώσεις DOM για να φέρει το DOM στην επιθυμητή κατάσταση.

Παρόλο που δημιουργούμε ένα στοιχείο που περιγράφει ολόκληρο το δέντρο διεπαφής χρήστη σε κάθε tick, μόνο ο κόμβος κειμένου του οποίου τα περιεχόμενα έχουν αλλάξει ενημερώνεται από το React DOM.

Εικονικό DOM

Ένα άλλο αξιοσημείωτο χαρακτηριστικό είναι η χρήση ενός εικονικού μοντέλου αντικειμένου εγγράφου ή εικονικού DOM. Το React δημιουργεί μια προσωρινή μνήμη δομής δεδομένων στη μνήμη, υπολογίζει

τις προκύπτουσες διαφορές και, στη συνέχεια, ενημερώνει αποτελεσματικά το εμφανιζόμενο DOM του προγράμματος περιήγησης. Επιτρέπει στον προγραμματιστή να γράψει κώδικα σαν να αποδίδεται ολόκληρη η σελίδα σε κάθε αλλαγή, ενώ οι βιβλιοθήκες του React αποδίδουν μόνο υποσυστήματα που αλλάζουν πραγματικά. Αυτή η επιλεκτική απόδοση παρέχει σημαντική αύξηση της απόδοσης, εξοικονομεί την προσπάθεια επαναπροσδιορισμού του στυλ CSS, της διάταξης και της απόδοσης για ολόκληρη τη σελίδα.

ES6

React ES6

codeburst.io, Brandon Morelli JavaScript — WTF is ES6, ES8, ES 2017, ECMAScript... ?

Το JavaScript ονομάστηκε αρχικά JavaScript με την ελπίδα να αξιοποιήσει την επιτυχία της Java. Στη συνέχεια, η Netscape υπέβαλε JavaScript στο ECMA International for Standardization. (Η ECMA είναι ένας οργανισμός που τυποποιεί τις πληροφορίες). Αυτό οδηγεί σε ένα νέο πρότυπο γλώσσας, γνωστό ως ECMAScript.

Με απλά λόγια, το ECMAScript είναι ένα πρότυπο. Ενώ η JavaScript είναι η πιο δημοφιλής εφαρμογή αυτού του προτύπου. Το JavaScript εφαρμόζει το ECMAScript και βασίζεται πάνω του.

Οπότε το ES6 σημαίνει ECMAScript 6, δηλ. η 6η έκδοση του ECMAScript που δημοσιεύθηκε το 2015.

React Components

Το συστατικό(Component) χωρίζει το περιβάλλον εργασίας χρήστη (UserInterface) σε ανεξάρτητα και επαναχρησιμοποιήσιμα τεμάχια που προαιρετικά δέχεται είσοδο τα props και επιστρέφει το στοιχείο του React. Ένα κουμπί, μια φόρμα, ένα παράθυρο διαλόγου, μια οθόνη: στις εφαρμογές React, όλα αυτά εκφράζονται συνήθως ως στοιχεία.

Υπάρχουν 2 τύποι components,

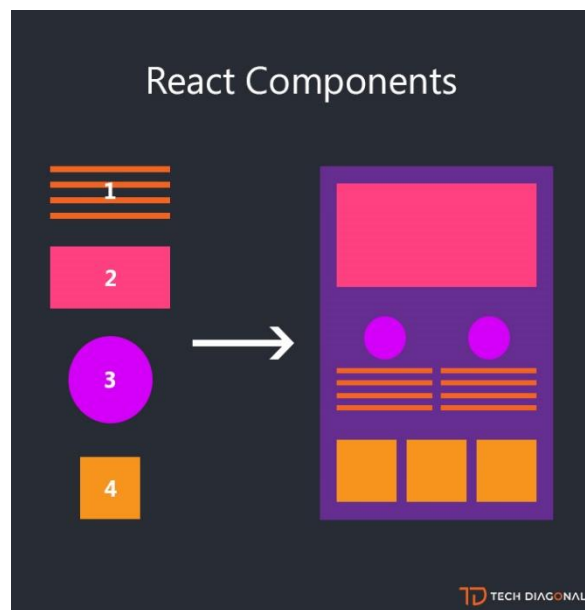
- ❖ Συστατικά συνάρτησης (Functional(stateless) Components)
- ❖ Class (stateful) Components

Ένα function component, είναι μια απλή συνάρτηση javascript που επιστρέφει ένα στοιχείο React,

```
function Welcome(props) {  
  return <h1>Hello, {props.name}</h1>; }  
}
```

Αυτή η συνάρτηση είναι ένα έγκυρο στοιχείο React επειδή δέχεται ένα μόνο όρισμα αντικειμένου "props" (που σημαίνει ιδιότητες) με δεδομένα και επιστρέφει ένα στοιχείο React. Τα στοιχεία αυτά ονομάζουμε «στοιχεία λειτουργίας» επειδή είναι κυριολεκτικά λειτουργίες JavaScript.

Αλλά εμείς χρησιμοποιήσαμε την δεύτερη κατηγορία συστατικών, την κλάση,



techdiagonal.com - Suhas Holla- Chapter 3 – Understanding ReactJS Components

```
export class MapContainer extends Component {
  constructor(props) {
    . . .
  }
  render() { . . . }
```

Διότι διατηρούν την τοπική κατάσταση του στοιχείου και διαχειρίζονται την κατάσταση του. Έχουν μεθόδους κύκλου ζωής που θα κληθούν όταν ένα στοιχείο προσαρτηθεί, ενημερωθεί και αποσυνδεθεί.

Στην ουσία το ReactDOM.render() καλεί ένα component με μία τιμή. Μετά η React καλεί το στοιχείο αυτό με την συγκεκριμένη τιμή ως props και έπειτα το στοιχείο επιστρέφει την τιμή αυτή σαν αποτέλεσμα στην οθόνη. Τέλος, το React DOM ενημερώνει το DOM ώστε να ταιριάζουν οι τιμές.

Επίσης, τα Components μπορεί να αναφέρεται σε άλλα στοιχεία(components) στην έξοδο τους.



React Google Maps

React Google Maps API Website by Ivan Konev

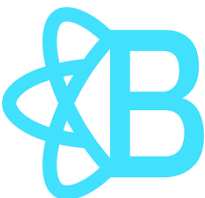
Το react-google-maps είναι η προσπάθεια της εφαρμογής του API χαρτών της google σε ένα προσωπικό έργο react.js.

Αυτό που προσφέρει το πακέτο αυτό είναι, ένα σύνολο στοιχείων του React που αναδιπλώνει τις συγκεκριμένες παρουσίες του Google Maps JavaScript API v3.

Το οποίο περιέχει στοιχεία για το UI του χάρτη, Markers, infobox, searchbar κλπ και κάνει τα εξής,

- props delegation
- events as callbacks
- lifecycle management (διαχείριση κύκλου ζωής)
- auto-mount on map (αυτόματη προσάρτηση στο χάρτη)

Τίποτα παραπάνω. Οπότε η διαδικασία για να εισαχθεί ο χάρτης στο πρόγραμμα θα περιγραφεί παρακάτω στην εξήγηση όλου του κώδικα.



React Bootstrap

react-bootstrap.github.io React Bootstrap

Το Bootstrap, η πιο δημοφιλής εργαλειοθήκη ανοιχτού κώδικα front-end στον κόσμο, με μεταβλητές Sass και mixins, σύστημα απόκρισης πλέγματος, εκτεταμένα προ εγκατεστημένα στοιχεία και ισχυρές προσθήκες JavaScript.

Έτσι και το React-Bootstrap αντικαθιστά το JavaScript Bootstrap. Κάθε στοιχείο έχει κατασκευαστεί από το μηδέν ως πραγματικό στοιχείο React, χωρίς περιττές εξαρτήσεις όπως το jQuery. Οι μέθοδοι και τα γεγονότα που χρησιμοποιούν το jQuery γίνονται επιτακτικά με χειρισμό του DOM. Αντίθετα, το React χρησιμοποιεί ενημερώσεις στην κατάσταση για την ενημέρωση του εικονικού DOM. Με αυτόν τον τρόπο, το React-Bootstrap παρέχει μια πιο αξιόπιστη λύση ενσωματώνοντας τη λειτουργία Bootstrap στο εικονικό DOM της React.

Ως μία από τις παλαιότερες βιβλιοθήκες του React, το React-Bootstrap εξελίχθηκε και αναπτύχθηκε παράλληλα με το React, καθιστώντας το μια εξαιρετική επιλογή ως το θεμέλιο διεπαφής χρήστη. Χτισμένο με γνώμονα τη συμβατότητα, αγκαλιάζουμε τον πυρήνα του bootstrap και προσπαθούμε να είμαστε συμβατοί με το μεγαλύτερο οικοσύστημα UI στον κόσμο. Βασιζόμενοι εξ ολοκλήρου στο φύλλο στυλ Bootstrap, το React-Bootstrap λειτουργεί μόνο με τα χιλιάδες θέματα Bootstrap που αγαπάτε ήδη.

Το μοντέλο συστατικών React μας δίνει περισσότερο έλεγχο στη μορφή και τη λειτουργία κάθε στοιχείου. Κάθε στοιχείο υλοποιείται έχοντας κατά νου την προσβασιμότητα. Το αποτέλεσμα είναι ένα σύνολο προσιτών από προεπιλογή στοιχείων, πέρα από ό, τι είναι δυνατό από το απλό Bootstrap.

Κάποια από τα στοιχεία αυτά είναι, alerts, buttons, modals, progressbar, paginations κλπ.



google images react confetti

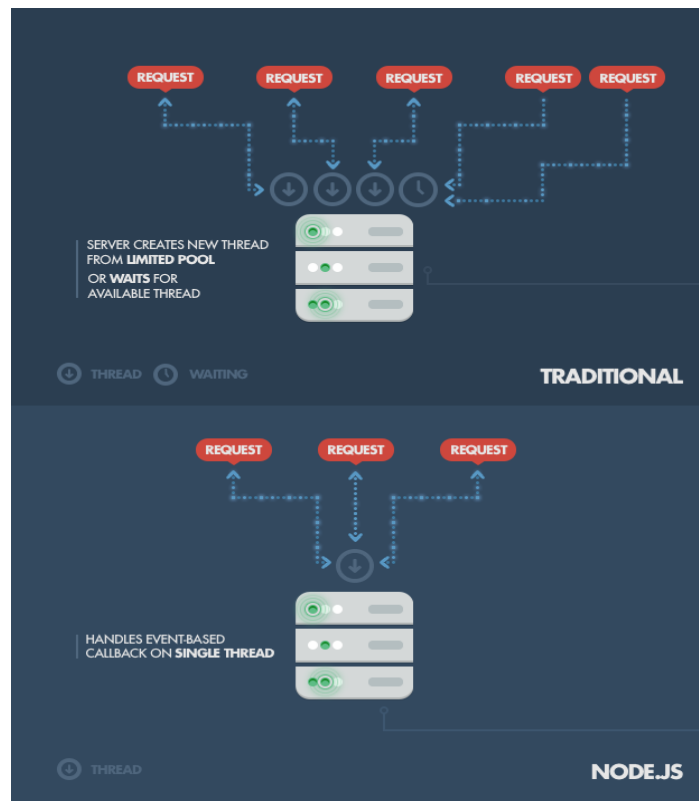
Ένα πακέτο του React κινουμένων σχεδίων (συγκεκριμένα κομφετί) για εύκολη προσθήκη κίνησης στο έργο.



Google Images-Wikimedia Commons File:Node.js logo.svg

Το Node.js, είναι ένα open-source περιβάλλον εκτέλεσης JavaScript που εκτελεί κώδικα έξω από ένα πρόγραμμα περιήγησης. Επιτρέπει στους προγραμματιστές να γράψουν εργαλεία γραμμής εντολών και για δέσμες ενεργειών διακομιστή, όπου εκτελώντας σκριπτάκια διακομιστή παράγεται το περιεχόμενο μιας δυναμικής ιστοσελίδας πριν από την αποστολή της στο πρόγραμμα περιήγησης του χρήστη. Είναι «ελαφρύ», δεν επιβαρύνει το σύστημα του υπολογιστή, γρήγορο και χρησιμοποιείται κυρίως για την γρήγορη κατασκευή κλιμακούμενων εφαρμογών διαδικτύου.

Όταν χρειάζεται να εκτελέσει λειτουργίες εισόδου/εξόδου, όπως το διάβασμα από το ίντερνετ, την πρόσβαση σε μία βάση δεδομένων ή το σύστημα αρχείων, αντί να αποκλείσει το νήμα και να χάσει τους κύκλους CPU που περιμένουν, θα επαναλάβει τις λειτουργίες όταν επανέλθει η απάντηση. Αυτό του επιτρέπει να χειρίζεται χιλιάδες ταυτόχρονες συνδέσεις με ένα μόνο διακομιστή.



Thread site Image, Node.js

Η διαφορά στην διαχείριση των αιτημάτων του με τα άλλα συστήματα είναι οι εξής,

- Τα PHP ή ASP :
 1. Στέλνουν την εργασία στο σύστημα αρχείων του υπολογιστή.
 2. Περιμένουν καθώς το σύστημα αρχείων ανοίγει και διαβάζει το αρχείο.
 3. Επιστρέφουν το περιεχόμενο στον πελάτη.
 4. Είναι έτοιμα να χειριστούν το επόμενο αίτημα.
- Node.js:
 1. Στέλνει την εργασία στο σύστημα αρχείων του υπολογιστή.
 2. Ετοιμάζεται να διαχειριστεί το επόμενο αίτημα.
 3. Όταν το σύστημα αρχείων έχει ανοίξει και διαβάσει το αρχείο, ο διακομιστής επιστρέφει το περιεχόμενο στον πελάτη.

Δηλαδή, το Node.js εξαλείφει την αναμονή και απλώς συνεχίζει με το επόμενο αίτημα. Εκτελεί έναν μη αποκλειστικό, ασύγχρονο προγραμματισμό, ο οποίος είναι πολύ αποδοτικός στη μνήμη. Επίσης, μπορεί να δημιουργήσει δυναμικό περιεχόμενο σελίδας, να ανοίξει/διαβάσει/γράψει/διαγράψει και να κλείσει αρχεία στον διακομιστή καθώς και να συλλέξει δεδομένα και να τα προσθέσει/διαγράψει/τροποποιήσει στην βάση δεδομένων.



Npm

Wikimedia Commons File:Npm-logo.svg - google images

Το npm (Node Package Manager) είναι ένα λογισμικό διαχείρισης πακέτων και εγκατάστασής τους. Το NPM έρχεται εξ ολοκλήρου μαζί με την εγκατάσταση του Node.js και παρέχει υποστήριξη διαχείρισης πακέτων και είναι δωρεάν στη χρήση του. Περιέχει γραμμή εντολών (CLI/Command Line Control) η οποία μπορεί να χρησιμοποιηθεί για το "κατέβασμα" και την εγκατάσταση λογισμικών. Επίσης, οι προγραμματιστές της JavaScript μπορούν εύκολα να ξαναχρησιμοποιήσουν τον κώδικα τους, να τον ενημερώσουν καθώς και να τον κοινοποιήσουν.

Κάποια από τα πράγματα που μπορεί να υποστηρίξει το npm είναι :

- Προσαρμογή των πακέτων του κώδικα για εφαρμογές, ή χρήση τους όπως είναι
- Άμεση αποθήκευση και χρήση των εργαλείων του
- Εκτέλεση πακέτων χωρίς την αποθήκευσή τους χρησιμοποιώντας το npx
- Δημιουργία orgs (organizations) για διατήρηση των πακέτων.
- Ενημέρωση των εφαρμογών εύκολα όταν ...ο κώδικας ενημερώνεται.
- Ανακάλυψη πολλαπλών τρόπων επίλυσης ενός προβλήματος. Όλα τα πακέτα καθορίζονται σε αρχεία που ονομάζονται «**package.json**». Το περιεχόμενο τους πρέπει να είναι γραμμένο σε

JSON αρχείο, όπου τουλάχιστον δύο πεδία πρέπει να είναι παρόντα στο αρχείο : name και version.

```
{
  "name": "corfu-escape-adventure",
  "version": "0.1.0",
  "private": true,
  "dependencies": {
    "google-map-react": "^1.1.4",
    "google-maps-react": "^2.0.2",
    "react": "^16.9.0",
    "react-bootstrap": "^1.0.0-beta.11",
    "react-dom": "^16.8.6",
    "react-google-maps": "^9.4.5",
    "react-scripts": "^3.4.0", . . .
  },
  "scripts": {
    "start": "react-scripts start",
    "build": "react-scripts build",
    "test": "react-scripts test", . . .
  }, . . .
}
```

Έτσι και στη εφαρμογή της παρούσας εργασίας στην αρχή δημιουργήθηκαν αυτόματα τα αρχεία αυτά και η μορφή τους είναι η εξής :

Για να τρέξει η εφαρμογή είναι απαραίτητο να ανοίξει πρώτα η γραμμή εντολών του Node.js , να κατευθυνθούμε στο αρχείο που είναι το παιχνίδι (cd file name) Και κατόπιν να γραφτεί η εντολή npm start.



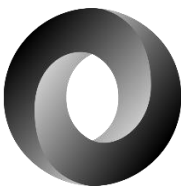
Npm start

Wikimedia Commons File:Npm-logo.svg - google images

Μπορεί το npm να διαχειρίζεται τα ληφθέντα αρχεία των απαραίτητων βιβλιοθηκών του έργου αλλά χρειάζεται και το Npm start για να ξεκινήσει το πακέτο που έχουμε εγκαταστήσει. Δηλαδή, εκτελεί μια αυθαίρετη εντολή που καθορίζεται στην ιδιότητα "start" του πακέτου του αντικειμένου "scripts" του. Εάν δεν έχει οριστεί η ιδιότητα "start" στο αντικείμενο "scripts" τότε θα εκτελεστεί το node server.js.

Για αυτό και για να ξεκινήσει η λειτουργία του προγράμματος τοπικά στον υπολογιστή είναι απαραίτητο να έχουν εγκατασταθεί όλες οι παραπάνω βιβλιοθήκες/ πακέτα καθώς και η γλώσσα react, στην οποία βασίστηκε η κατασκευή του προγράμματος της εργασίας. Στο τερματικό του Node.js προσδιορίζουμε ποιο πρόγραμμα θα εκτελεστεί και με την εντολή npm start εκτελείτε το παιχνίδι.

```
C:\Users\mirto>cd corfu-escape-adventure  
C:\Users\mirto\corfu-escape-adventure>npm start
```



Json

Figure 1 freebiesupply.com json logo

Το JSON (JavaScript Object Notation) είναι μια ελαφριά μορφή ανταλλαγής δεδομένων. Βασίζεται σε ένα υποσύνολο του Προγράμματος Γλώσσας Προγραμματισμού JavaScript ECMA-262 3η Έκδοση - Δεκέμβριος 1999.

Το JSON είναι ένα πρότυπο κειμένου το οποίο είναι τελείως ανεξάρτητο από γλώσσες προγραμματισμού αλλά χρησιμοποιεί πρακτικές (conventions) οι οποίες είναι γνωστές στους προγραμματιστές της οικογένειας προγραμματισμού C, συμπεριλαμβανομένων των C, C++, C#, Java, JavaScript, Python και πολλών άλλων. Αυτές οι ιδιότητες κάνουν το JSON μια ιδανική γλώσσα προγραμματισμού ανταλλαγής δεδομένων.

Το JSON είναι χτισμένο σε δύο δομές:

- Μια συλλογή από ζευγάρια ονομάτων/τιμών. Σε διάφορες γλώσσες προγραμματισμού, αυτό αντιλαμβάνεται ως ένα object ("αντικείμενο"), καταχώριση, δομή, λεξικό, πίνακα hash (hash table), λίστα κλειδιών, ή associative πίνακα.
- Μία ταξινομημένη λίστα τιμών. Στις περισσότερες γλώσσες προγραμματισμού, αυτό αντιλαμβάνεται ως ένας πίνακας (array), διάνυσμα, λίστα, ή ακολουθία.

Το δεύτερο χρησιμοποιήθηκε στον κώδικα (λίστα τιμών) και αυτά είναι τα universal data structures. Ουσιαστικά όλες οι μοντέρνες γλώσσες προγραμματισμού τα υποστηρίζουν με τον έναν ή τον άλλον τρόπο. Λογικό είναι πως ένα πρότυπο δεδομένων το οποίο είναι εύκολα μεταβαλλόμενο με γλώσσες προγραμματισμού, οι οποίες επίσης είναι βασισμένες σε αυτές τις δομές.

Στο JSON αρχείο, παίρνουν αυτές τις μορφές :

Ένας πίνακας (array) είναι μια συλλογή από τιμές σε σειρά. Ένας πίνακας (array- «museumInfo») ξεκινάει με [αριστερή αγκύλη και τελειώνει με]δεξιά αγκύλη. Οι τιμές χωρίζονται με ,κόμμα.

Μία τιμή μπορεί να είναι string μέσα σε διπλά quotes, ή αριθμός (number), ή **true** ή **false** ή **null**, ή αντικείμενο (object) ή πίνακας (array). Αυτές οι τιμές μπορεί να είναι και ανακατεμένες ("title", "information").

Έτσι καταλήγουμε να έχουμε το εξής αποτέλεσμα,

```
{
  "museumInfo": [
    {
      "id": "1",
      "title": "Σινοϊαπωνικό Μουσείο",
      "information": [ "Στα Ανάκτορα των Αρχαγγέλου Μιχαήλ . . ." ] }, . . .
  ]
}
```

Πίνακες που περιέχουν πληροφορία , όπου το κάθε στοιχείο είναι μοναδικό. Σαν το id που αναφέρθηκε παραπάνω (getElementById στο ReactDOM).



VS CODE

google images - <https://worldvectorlogo.com/logo/visual-studio-code-1>

Το Visual Studio Code είναι ένα ελαφρύ αλλά ισχυρό πρόγραμμα επεξεργασίας πηγαίου κώδικα που λειτουργεί στην επιφάνεια εργασίας και έρχεται με ενσωματωμένη υποστήριξη για JavaScript, TypeScript και Node.js και διαθέτει πλούσιο οικοσύστημα επεκτάσεων για άλλες γλώσσες (όπως React, C #, Java, Python, PHP κλπ.) και ώρες εκτέλεσης (όπως .NET και Unity).

Το Visual Studio Code συνδυάζει την απλότητα ενός επεξεργαστή πηγαίου κώδικα με ισχυρά εργαλεία προγραμματιστή, όπως ολοκλήρωση κώδικα και εντοπισμός σφαλμάτων IntelliSense ώστε για να μην χάνουν χρόνο οι προγραμματιστές. Δίνει την δυνατότητα για έξυπνη ολοκλήρωση κώδικα, γρήγορη, ισχυρή επεξεργασία, πλοήγηση και αναπαράσταση κώδικα. Επιτρέπει την ενεργοποίηση πρόσθετων γλωσσών, θέματα, έξυπνες συντομεύσεις πληκτρολογίου, εγχειρίδια χρήσης και step by step tutorials. Διαθέτει έναν γρήγορο επεξεργαστή πηγαίου κώδικα, ιδανικό για καθημερινή χρήση και ο κώδικας VS ενσωματώνεται με εργαλεία κατασκευής και δέσμης ενεργειών για την εκτέλεση κοινών εργασιών, με υποστήριξη για το Git και με ένα έργο ανοιχτού κώδικα, ώστε να παρέχεται η συμβολή στην αναπτυσσόμενη και ζωντανή κοινότητα του GitHub.

Ο κώδικας VS περιλαμβάνει εμπλουτισμένη ενσωματωμένη υποστήριξη για ανάπτυξη Node.js με JavaScript και TypeScript, που υποστηρίζεται από τις ίδιες υποκείμενες τεχνολογίες που οδηγούν το Visual Studio. Ο κώδικας VS περιλαμβάνει επίσης εξαιρετικά εργαλεία για τεχνολογίες Ιστού όπως JSX / React, HTML, CSS, SCSS, Less και JSON. Χρησιμοποιώντας το Electron, συνδυάζει τεχνολογίες ιστού όπως το JavaScript και το Node.js με την ταχύτητα και την ευελιξία των εγγενών εφαρμογών.

ΥΛΟΠΟΙΗΣΗ

Για το πρακτικό κομμάτι της εργασίας ήταν απαραίτητη η χρήση των αναφερόμενων εργαλείων του προηγούμενου κεφαλαίου, όπως πλατφόρμες για ανάπτυξη κώδικα και γλώσσες προγραμματισμού.

Η γλώσσα προγραμματισμού που χρησιμοποιήθηκε για την ανάπτυξη του προγράμματος ήταν η React της Facebook, η οποία καθόρισε ποια εργαλεία χρειάστηκαν για την υλοποίηση του παιχνιδιού.



React

React Logo from official site

Το React ή αλλιώς React Js είναι μια δηλωτική, αποτελεσματική και ευέλικτη βιβλιοθήκη JavaScript για τη δημιουργία διεπαφών χρήστη. Επιτρέπει την σύνθεση σε σύνθετα περιβάλλοντα χρήστη από μικρά και απομονωμένα κομμάτια κώδικα που ονομάζονται "στοιχεία" ("components"). Δημιουργήθηκε για πρώτη φορά από τον Jordan Walke, έναν μηχανικό λογισμικού που εργάζεται στο Facebook. Το React κυκλοφόρησε για πρώτη φορά στο newsfeed του Facebook το 2011 και στο Instagram.com το 2012.

Η χρήση του React εφαρμόζεται ειδικά για εφαρμογές μιας σελίδας επιτρέποντας την δημιουργία επαναχρησιμοποιήσιμων στοιχείων διεπαφής χρήστη και έτσι οι προγραμματιστές δημιουργούν μεγάλες εφαρμογές ιστού που μπορούν να αλλάζουν δεδομένα, χωρίς να παραφορτώνουν τη σελίδα.

Η επιλογή αυτής της βιβλιοθήκης για την ανάπτυξη της εργασίας έγινε για τους εξής λόγους,

Αρχικά το ReactJS είναι απλούστερο στην άμεση κατανόηση. Η προσέγγιση με βάση τα συστατικά, ο σαφώς καθορισμένος κύκλος ζωής και η χρήση απλού JavaScript το καθιστούν πολύ απλό για εκμάθηση και εκμάθηση δημιουργίας ενός επαγγελματικού ιστού (και εφαρμογές για κινητά) και την υποστήριξη του. Χρησιμοποιεί μια ειδική σύνταξη που ονομάζεται JSX που επιτρέπει την ανάμιξη HTML με JavaScript. Αυτό δεν είναι απαραίτητο, ο προγραμματιστής μπορεί ακόμα να γράψει σε απλή JavaScript, αλλά το JSX είναι πολύ πιο εύκολο στη χρήση.

Επίσης, είναι εύκολη βιβλιοθήκη στην εκμάθηση. Όποιος έχει βασικές προηγούμενες γνώσεις στον προγραμματισμό μπορεί εύκολα να καταλάβει το React ενώ το Angular και το Ember αναφέρονται ως «Γλώσσα για συγκεκριμένους τομείς», υπονοώντας ότι είναι δύσκολα στη μάθηση. Για την συγκεκριμένη γλώσσα, χρειάζονται βασικές γνώσεις CSS και HTML.

Όσον αφορά τα δεδομένα, χρησιμοποιεί μονόδρομη σύνδεση δεδομένων και μια αρχιτεκτονική εφαρμογών που ονομάζεται Flux. Αυτή ελέγχει τη ροή δεδομένων σε στοιχεία μέσω ενός σημείου ελέγχου - του διεκπεραιωτή. Είναι πιο εύκολο στον εντοπισμό αυτοδύναμων στοιχείων μεγάλων εφαρμογών ReactJS. Το πιο σημαντικό είναι η δοκιμή των αναπτυσσόμενων εφαρμογών. Με την ReactJS είναι εξαιρετικά εύκολο. Οι προβολές αντιδράσεων μπορούν να αντιμετωπίζονται ως συναρτήσεις της κατάστασης, οπότε μπορούμε να χειριστούμε με την κατάσταση που περνάμε στην προβολή ReactJS και να ρίξουμε μια ματιά στην έξοδο και να ενεργοποιήσει ενέργειες, συμβάντα, συναρτήσεις κ.λπ.

Η React μπορεί να χρησιμοποιήσει και άλλες βιβλιοθήκες όπως το Nodejs. Τόσο το Nodejs όσο και το React είναι γλώσσες javascript που μπορούν να εκτελεστούν τόσο από τον πελάτη όσο και από τον

διακομιστή. Οι προγραμματιστές μπορούν να εκτελέσουν τον κώδικα Reactjs απευθείας στο περιβάλλον Nodejs.

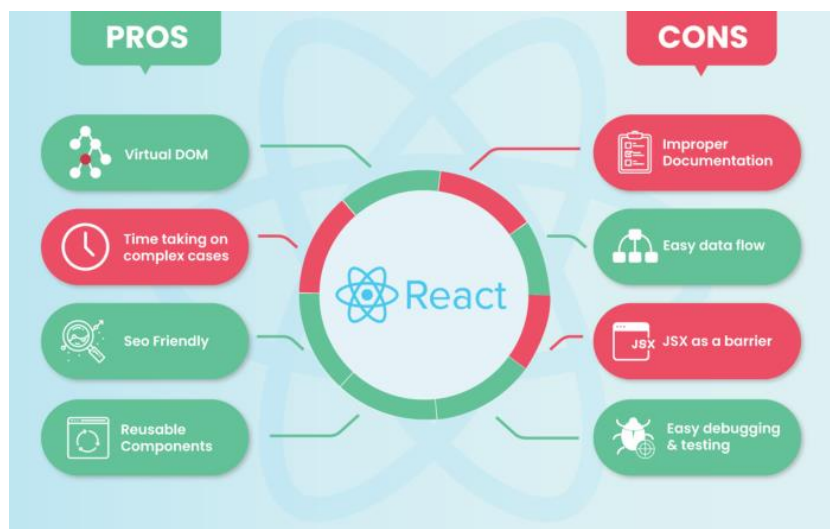
Στην πραγματικότητα, το Nodejs είναι η πιο βολική πλατφόρμα για φιλοξενία καθώς και για τη λειτουργία ενός διακομιστή Ιστού για μια εφαρμογή React. Αυτό οφείλεται σε δύο βασικούς λόγους: Χρησιμοποιώντας ένα NPM (Node Package Manager), το Node συνεργάζεται με το μητρώο NPM για την εύκολη εγκατάσταση οποιουδήποτε πακέτου μέσω του NPM CLI. Ο κόμβος ομαδοποιεί μια εφαρμογή React σε ένα μόνο αρχείο για εύκολη συλλογή χρησιμοποιώντας το webpack και πολλές άλλες ενότητες Node. Το Nodejs χρησιμοποιεί έναν κινητήρα V8 βελτιστοποιημένο για την ταχύτητα για την αντιμετώπιση μαζικών αιτημάτων που διεκπεραιώνονται μέσω λειτουργιών επανάκλησης για διασφάλιση ποιότητας και ποσότητας. Το React DOM διαθέτει στοιχεία ειδικά σχεδιασμένα για να δουλεύουν με Nodejs που μειώνουν τις γραμμές κώδικα, καθιστώντας την απόδοση διακομιστή συγκριτικά εύκολη.

Μπορούμε ακόμα να χρησιμοποιήσουμε το Nodejs για να εκτελέσουμε τον διακομιστή ιστού που φιλοξενεί την εφαρμογή React. Βέβαια, δεν είναι απαραίτητη η χρήση του Nodejs ως backend , κάθε φορά για την χρήση του React.

Ωστόσο, στην εργασία αυτή έγινε η χρήση του, διότι το Reactjs, που είναι ένα frontend framework, φορτώνει τα εξαρτήματα που συνδέονται με τις συγκεκριμένες διαδρομές. Ενώ το Nodejs, το πλαίσιο backend δεν ενδιαφέρεται για το ποιο στοιχείο φορτώνεται σε ένα συγκεκριμένο αίτημα διεύθυνσης URL, παρέχει μόνο ένα περιβάλλον για ολόκληρη την επικοινωνία δεδομένων.

Είναι απαραίτητο όταν η εφαρμογή ιστού χρειάζεται χειρισμό πολλαπλών αιτημάτων και διατήρηση της ισορροπίας φόρτωσης διακομιστή σε πραγματικό χρόνο(δηλ. φορτώνει συνεχώς τις πληροφορίες για τον χάρτη μας). Η δημιουργία API JSON για την εφαρμογή είναι πολύ αποτελεσματική με τα Nodejs λόγω του επαναχρησιμοποιήσιμου κώδικα και της εύκολης κοινής χρήσης κώδικα(api json , που καλείται ο χάρτης). Τέλος, χρειάζεται για ανάπτυξη εφαρμογών μίας σελίδας στο React κατά τη χρήση του κόμβου για τη δημιουργία ενός ελαφρού μοντέλου backend για ασύγχρονη φόρτωση δεδομένων μέσω λειτουργιών επανάκλησης (ο χάρτης μας είναι μια σελίδα).

Επίσης ως προς την δημοτικότητα του σήμερα, το ReactJS έχει γίνει πολύ δημοφιλές λόγω της επιπλέον απλότητας και ευελιξίας του. Πολλοί άνθρωποι το αναφέρουν ως το μέλλον της ανάπτυξης ιστού. Υπολογίζεται ότι περισσότεροι από 1.300 προγραμματιστές και πάνω από 94.000 ιστότοποι χρησιμοποιούν το ReactJS (σύμφωνα με το άρθρο "Why ReactJS is gaining so much popularity these days", Thinkwik, 2017, medium.com). Μέρος αυτής της τεράστιας δημοτικότητας προέρχεται από το γεγονός ότι κορυφαίες εταιρείες όπως το Facebook, το PayPal, το Uber, το Instagram και η Airbnb το χρησιμοποιούν για την επίλυση ζητημάτων που σχετίζονται με το περιβάλλον εργασίας χρήστη.



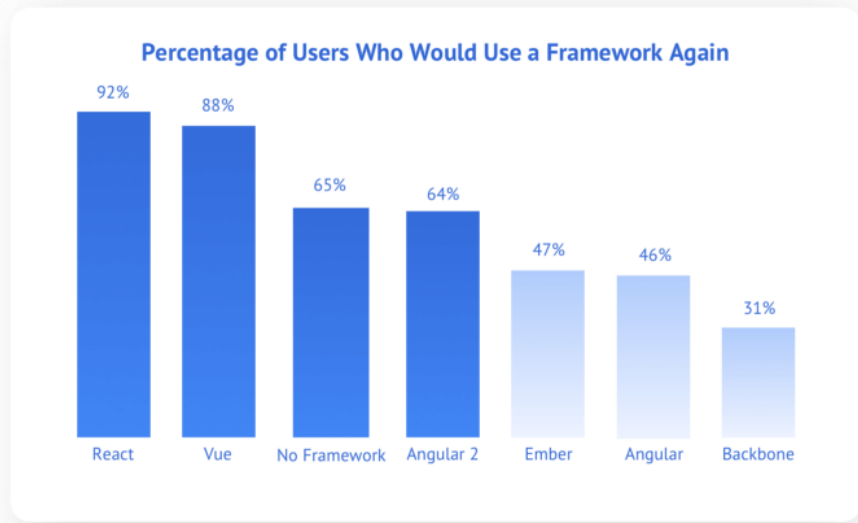
"Why ReactJS is gaining so much popularity these days", Thinkwik, 2017, medium.com

Οι περισσότεροι προγραμματιστές αναγκάστηκαν να κάνουν πολλή κωδικοποίηση και έπρεπε να επεξεργαστούν ξανά τους περισσότερους κωδικούς ακόμα και όταν δημιουργούν στοιχεία που αλλάζουν συχνά. Αυτό που ήθελαν ήταν ένα πλαίσιο που θα τους επέτρεπε να αναλύσουν σύνθετα στοιχεία και να επαναχρησιμοποιήσουν τους κωδικούς για να ολοκληρώσουν τα έργα τους πιο γρήγορα έτσι και εμείς διαλέξαμε αυτή την βιβλιοθήκη για την εργασία μας. Το ReactJS χρησιμοποιεί το JSX, ένα απλό JavaScript που επιτρέπει την προσφορά HTML και χρησιμοποιεί αυτές τις σύνθετες ετικέτες HTML για την απόδοση δευτερευόντων στοιχείων. Η σύνταξη HTML επεξεργάζεται σε κλήσεις JavaScript του React Framework. Είναι δυνατή και η συγγραφή σε καθαρή παλιά JavaScript.

Επιτρέπει συγγραφή εφαρμογών μέσω JavaScript και το JSX είναι ένα από τα καλύτερα χαρακτηριστικά που έχει κάνει το ReactJS εύκολο αλλά και διασκεδαστικό. Οι προγραμματιστές μπορούν εύκολα να δημιουργήσουν μια νέα δυνατότητα διεπαφής χρήστη και να τη δουν να εμφανίζεται σε πραγματικό χρόνο. Φέρνει HTML απευθείας στο JS και καθιστά δυνατή τη διάσπαση της σύνθετης ανάπτυξης UI / UX σε απλούστερα στοιχεία. Αυτό είναι σημαντικό ώστε να γίνει το κάθε στοιχείο πιο διαισθητικό.

Ένα από τα θετικά είναι ότι χρησιμοποιεί το Virtual DOM που κάνει την εφαρμογή γρήγορη και βοηθάει το ReactJS να γνωρίζει πότε ακριβώς θα γίνει εκ νέου απόδοση ή πότε θα αγνοήσει ορισμένα συγκεκριμένα κομμάτια του DOM επειδή μπορεί να ανιχνεύσει πότε άλλαξαν τα δεδομένα. Ένα περιβάλλον εργασίας χρήστη που αντιδρά άμεσα είναι ζωτικής σημασίας για τη βελτίωση της εμπειρίας του χρήστη.

Μια σύγκριση του ReactJS με άλλα δημοφιλή πλαίσια, frameworks.



"Why ReactJS is gaining so much popularity these days", Thinkwik, 2017, medium.com

Οι εξής παράγοντες είναι που καθόρισαν την σύγκριση αυτή,

1. Απόδοση χρόνου εκτέλεσης: Τόσο το ReactJS όσο και το VueJS είναι εξαιρετικά για την ταχύτητα. Στο ReactJS, κάθε φορά που αλλάζει ένα στοιχείο, ενεργοποιείται μια εκ νέου απόδοση για ολόκληρο το δευτερεύον δέντρο. Εάν δεν θέλετε να κάνετε εκ νέου απόδοση, σκεφτείτε να χρησιμοποιήσετε το PureComponent. Στο VueJS, οι εξαρτήσεις των στοιχείων παρακολουθούνται αυτόματα κατά την απόδοση. Αυτό σημαίνει ότι το σύστημα γνωρίζει τα στοιχεία που απαιτούν εκ νέου απόδοση μετά την αλλαγή της κατάστασης.
2. Εφαρμογή HTML και CSS: Τα πάντα στο ReactJS είναι JS. Κάθε δομή HTML εκφράζεται μέσω JSX, ενώ οι τελευταίες τάσεις προσθέτουν διαχείριση CSS και στο JS. Ωστόσο, το VueJS αγκαλιάζει τις κλασικές τεχνολογίες ιστού και τις βελτιώνει.
3. Απόδοση χρόνου εκτέλεσης: Στην απόδοση, η αξιολόγηση Angular vs React αποκαλύπτει ότι το ReactJS αποδίδει καλύτερη απόδοση. Το AngularJS γίνεται σχετικά αργό όταν υπάρχουν πάρα πολλοί χρήστες επειδή αλλάζει οτιδήποτε στο πεδίο εφαρμογής του. Εκτός αυτού, ο κύκλος χώνευσης στο AngularJS απαιτεί εκτέλεση πολλές φορές πριν σταθεροποιηθεί εάν ένας χρήστης ενεργοποιήσει μια ενημέρωση. Το ReactJS δεν αντιμετωπίζει αυτό το ζήτημα επειδή έχει ένα σύστημα παρακολούθησης εξάρτησης που χρησιμοποιεί μια ουρά ασύγχρονης.
4. Το βήμα εκμάθησης: Για να ξεκινήσει ένας νέος προγραμματιστής με το ReactJS, το βήμα εκμάθησης είναι λιγότερο απότομο σε σύγκριση με το AngularJS. Κάποιος πρέπει μόνο να εξοικειωθεί με το JS και το HTML στην περίπτωση του ReactJS. Ωστόσο, η εκμάθηση του AngularJS είναι μια μακρά διαδικασία που απαιτεί την τήρηση πάρα πολλών εννοιών.

Από τη σύγκριση, το ReactJS βγαίνει ως ένα ισχυρότερο πλαίσιο λόγω της δυνατότητας διάσπασης του σύνθετου περιβάλλοντος χρήστη και επιτρέπει στους χρήστες να εργάζονται σε μεμονωμένα στοιχεία.

Το μέλλον του ReactJS είναι σίγουρο. Το Facebook και ολόκληρη η ομάδα του ReactJS έχουν δείξει τη δέσμευσή τους για ενίσχυση της αποτελεσματικότητας του ReactJS.

ΒΗΜΑΤΑ ΥΛΟΠΟΙΗΣΗΣ

Ο κώδικας

Αρχικά για να δημιουργηθεί η εφαρμογή τοπικά στον υπολογιστή μπορούμε να κατεβάσουμε το Node.js και να χρησιμοποιήσουμε το τερματικό του. Γίνεται η λήψη αυτού του περιβάλλοντος JavaScript και η λήψη ενός διαχειριστή πακέτων, ώστε να μπορούμε να κατεβάζουμε και να εγκαθιστούμε βιβλιοθήκες, συμπεριλαμβανομένου του React.

Στη συνέχεια, χρησιμοποιούμε τη γραμμή εντολών (CLI) για να αλληλοεπιδράσουμε με το Node.js και το Node Package Manager (NPM) για να εγκαταστήσουμε το React και γράφουμε τις εξής εντολές,

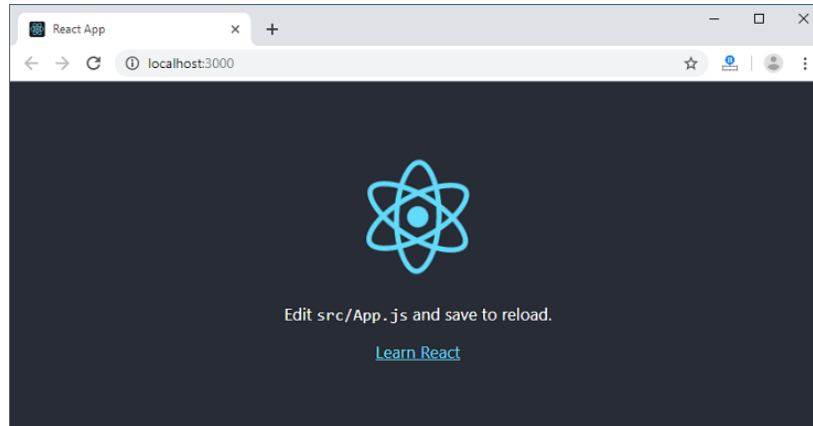
```
mkdir corfu-escape-adventure  
npm install -g create-react-app  
create-react-app corfu-escape-adventure  
cd corfu-escape-adventure  
npm start
```

Με αυτές τις εντολές λέμε να δημιουργηθεί πρώτα νέος κατάλογος στο σύστημα μας που ονομάζεται “corfu-escape-adventure” και έπειτα εγκαθιστά την ενότητα “create-react-app” που καθιστά εύκολη τη δημιουργία και την ανάπτυξη του React project μας με μία μόνο εντολή. Έπειτα φτιάχνουμε την εφαρμογή μας με όνομα “corfu-escape-adventure” και τρέχουμε το project μας. Δηλαδή λέμε cd “corfu-escape-adventure” (η εντολή θα μας μεταφέρει στο φάκελο που θα αναπτυχθεί η εφαρμογή) και npm start για να ξεκινήσει το πρόγραμμα περιήγησης στη θύρα 3000.

Το React ανοίγει το Localhost:3000, ο αριθμός θύρας στην οποία θα επικοινωνήσει ο ιστότοπος που φιλοξενείται τοπικά στο μηχάνημά μας. Το μήνυμα που λαμβάνουμε την γραμμή εντολών,

```
Compiled successfully!  
  
You can now view corfu-escape-adventure in the browser.  
  
Local : http://localhost:3000  
  
On Your Network: http://number.number.1.6:3000
```

Έτσι εμφανίζεται η εφαρμογή μας στο πρόγραμμα περιήγησης με την αρχική καθορισμένη εμφάνιση της κάθε React εφαρμογής.



Οπότε είναι δυνατόν να ξεκινήσουμε να χτίζουμε τον κώδικα στο Vscode πρόγραμμα. Το μόνο που κάνουμε σε αυτό το πρόγραμμα επεξεργασίας κώδικα είναι να ανοίξουμε τον σωστό φάκελο που περιέχει το project μας από τον φάκελο που δημιουργήσαμε νωρίτερα, (corfu-escape-adventure) και να κάνουμε τις κατάλληλες αλλαγές για να «χτιστεί» η επιθυμητή εφαρμογή.

Όταν ξεκινάει ένα React project υπάρχουν από την αρχή κάποια αρχεία. Αυτά τα αρχεία μπορούμε είτε να τα σβήσουμε είτε να τα τροποποιήσουμε όπως επιθυμούμε.

```
Corfu-escape-adventure/
├── README.md
├── node_modules/
├── package.json
├── public/
│   ├── index.html
│   └── favicon.ico
└── src/
    ├── App.css
    ├── App.js
    ├── App.test.js
    ├── index.css
    ├── index.js
    ├── logo.svg
    └── serviceWorker.js
```

Αυτά τα αρχεία πρέπει να υπάρχουν με την ακριβή ονομασία τους,

public / index.html , είναι το πρότυπο σελίδας και το src / index.js είναι το σημείο εισόδου της JavaScript.

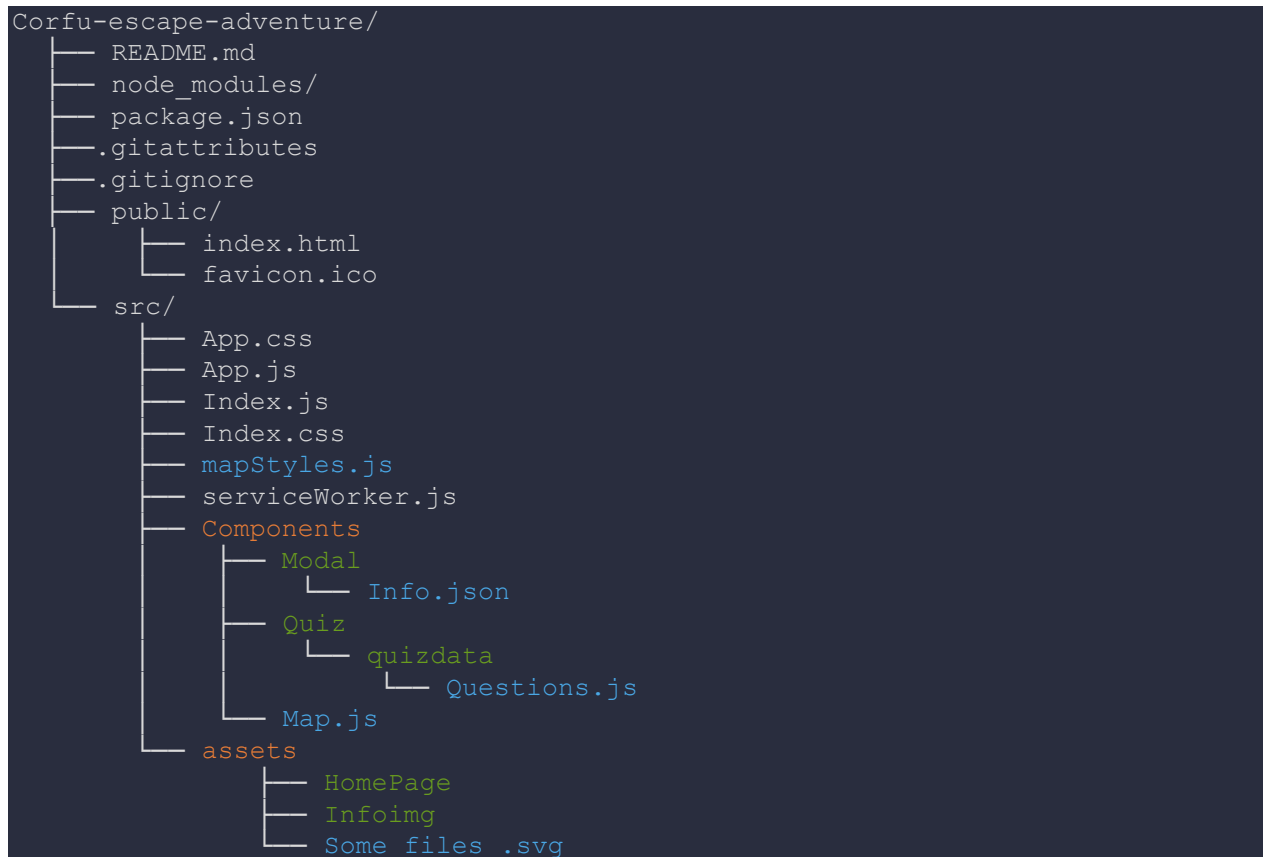
Το αρχείο serviceWorker, είναι προαιρετικός κώδικας και χρησιμοποιείται για την εγγραφή ενός “εργαζομένου”. Αυτό επιτρέπει στην εφαρμογή να φορτώνει γρηγορότερα σε επόμενες επισκέψεις στην παραγωγή και δίνει δυνατότητες εκτός σύνδεσης.

Τα στατικά αρχεία μας βρίσκονται στον δημόσιο κατάλογο. Τα αρχεία σε αυτόν τον κατάλογο θα διατηρήσουν το ίδιο όνομα κατά την ανάπτυξη στην παραγωγή. Έτσι, μπορούν να αποθηκευτούν στην κρυφή μνήμη από την πλευρά του πελάτη και να βελτιώσουν τους συνολικούς χρόνους λήψης.

Όλα τα δυναμικά στοιχεία θα βρίσκονται στο src. Για να διασφαλιστεί ότι, από την πλευρά του πελάτη, λαμβάνεται μόνο η πιο πρόσφατη έκδοση και όχι το προσωρινά αποθηκευμένο αντίγραφο,

Στην περίπτωση μας δεν πειράξαμε τον Public φάκελο και έγινε αλλαγή μόνο στον src φάκελο. Συγκεκριμένα έγιναν προσθήκες δύο φακέλων «components» και «assets», όπου ο πρώτος περιέχει άλλα προγράμματα που θα καλέσει η εφαρμογή μας και ο δεύτερος φάκελος περιέχει όλες τις εικόνες που χρησιμοποιήθηκαν.

Οπότε τώρα η παραπάνω δομή αλλάζει, οι κύριοι φάκελοι είναι με πορτοκαλί, οι δευτερεύοντες με πράσινο και με μπλε τα αρχεία τους. Το κάθε χρώμα υποδηλώνει και μία διαφορετική κατηγορία χρήσης των αρχείων αυτών.

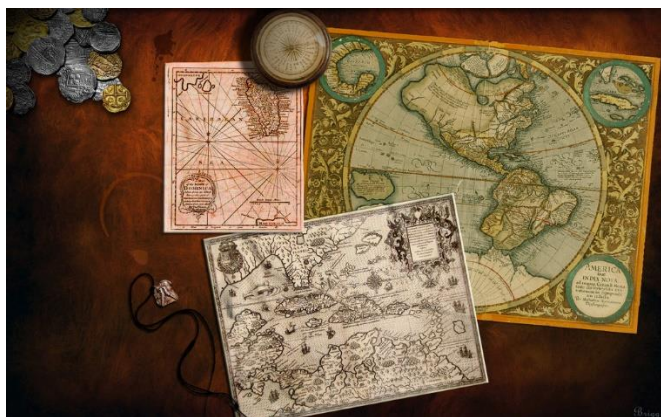


Θα εξηγήσουμε με την σειρά που είναι το σχέδιο τι κάνει το κάθε αρχείο.

Βρισκόμαστε στον φάκελο src, και το πρώτο αρχείο που βλέπουμε είναι το αρχείο «App.css». Αυτό το αρχείο είναι γραμμένο σε γλώσσα CSS και «φροντίζει» την εμφάνιση του κάθε στοιχείου και το πώς θα εμφανίζεται στην οθόνη του κάθε χρήστη.

Επειδή η ιστορία του παιχνιδιού έχει να κάνει με πειρατές που θέλουν να αποτρέψουν τον παίκτη να φύγει από το νησί έγινε επιλογή του φόντου, του αρχικού μενού να είναι ένα τραπέζι που επάνω του υπάρχουν νομίσματα και χάρτες, όπως τα πειρατικά καράβια.

Έγινε έρευνα στο ίντερνετ και μέσα από το Google Images βρέθηκαν οι αρχικές εικόνες. Δηλαδή ένας χάρτης τραπεχιού με πειρατικούς χάρτες και νομίσματα και ένας Ελληνικός πειρατικός χάρτης.



Έγινε η ένωση τους στο δωρεάν σαιτ που υπάρχει στο ίντερνετ, το pixlr.com και το επιθυμητό φόντο μας είναι,



Σε αυτό το πρόγραμμα επίσης έγινε και η δημιουργία του λόγκο, του ονόματος του παιχνιδιού.



Οπότε για να εμφανιστεί η παραπάνω εικόνα και το λόγκο μαζί στην αρχική σελίδα του προγράμματος μας το διαμορφώνουμε στο css αρχείο σαν μια εικόνα στατική.

```
.Menu {
  background-image: url(../assets/HomePage/Pirate-Wallpaper1.jpg);
  /*To stay in the background*/
  z-index: 3;
  /* Set up proportionate scaling */
  width: 100%;
  /* Set up positioning to static in order not to change*/
  position: static;
  /* top: 0;
  left: 0; */
  /*Let all the flexible items be the same length, regardless of its content*/
```

```

flex: 1;
/*A background-image that will not scroll with the page (fixed):*/
background-attachment: fixed;
}
.Logo {
  z-index: 4; /*so that it will be upfront of the other images*/
  position: absolute; /*it doesnt move to the page*/
  padding: 20px;
  /*distance from the bottom of the page upwards*/
  margin-left: 820px; /*distance from the left of the page*/
  margin-right: auto; /*distance from the right of the page*/
  margin-top: 90px;
  /*Let all the flexible items be the same length, regardless of its content*/
  flex: 1;
  /*A background-image that will not scroll with the page (fixed):*/
  background-attachment: fixed;}

```

Τα τρία κουμπιά, “play” “story” και “credits”, όπου για το καθένα από αυτά δηλώνουμε το σχήμα τους (στρογγυλό), το χρώμα τους και όταν τα πατάει ο χρήστης να αλλάζουν χρώμα(.hover και .activate).

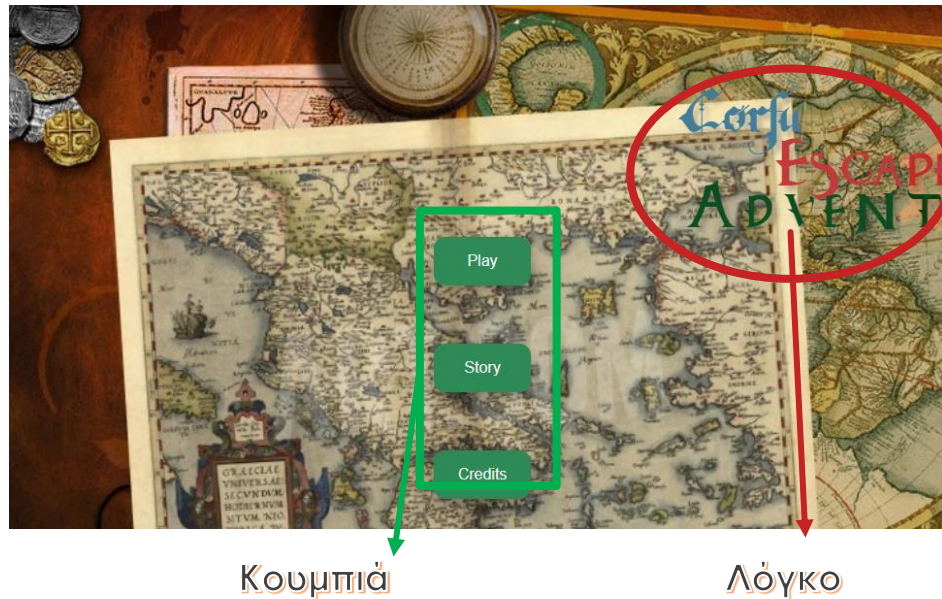
```

.Play_button {
  display: table-cell;
  z-index: 4; /*so that it will be upfront of the other images*/
  position: absolute; /*it doesnt move to the page*/
  margin-top: -560px; /*distance from the top of the page */
  margin-left: 600px; /*distance from the left of the page*/
  width: 100px;
  height: 50px;
  font-size: 16px; /*Button Size*/
  text-align: center;
  text-decoration: none;
  color: #f5f5f5; /*text color*/
  background-color: #2e8b57;
  border-radius: 12px; /*property to add rounded corners to a button*/
  border: #f5deb3; /*border color*/
  transition-duration: 0.4s;
  cursor: pointer;
  box-shadow: 0 8px 16px 0 rgba(0, 0, 0, 0.2), 0 6px 20px 0 rgba(0, 0, 0, 0.19);
}
.Play_button:hover {
  background-color: #008080;
  color: white;
  border: 2px solid #deb887;}
.Play_button:active {
  background-color: #3e8e41;
  box-shadow: 0 5px #666;
  transform: translateY(4px);}

```

Αντίστοιχα και για τα άλλα δύο είναι περίπου ο ίδιος κώδικας.

Οπότε το αποτέλεσμα που προκύπτει είναι,



Μετά έχουμε την εμφάνιση των Modals και τον τρόπο που θα εμφανίζονται. Το modal είναι ένα πλαίσιο μηνυμάτων που εμφανίζεται στο πάνω μέρος της οθόνης. Τα modals τοποθετούν μια επικάλυψη στην οθόνη, επομένως, έχουν οπτική προτεραιότητα έναντι όλων των άλλων στοιχείων.

Φτιάχνουμε την απλή εμφάνιση του και

```
.modal {  
  position: absolute; /*in order not to move to the page*/  
  z-index: 4; /*so that it will be upfront of the other images*/  
  top: 0;  
  left: 0;  
  width: 100%;  
  height: 100%;  
  background: rgba(0, 0, 0, 0.6);  
  transition: all ease 0.8s; /*movement*/  
}
```

Δηλώνουμε την επικεφαλίδα του,

```
.modal-header {  
  background: var(--modal-color);  
  height: -340px;  
  padding: 0px;  
  color: #fff;  
  border-top-left-radius: 5px;  
  border-top-right-radius: 5px;}
```

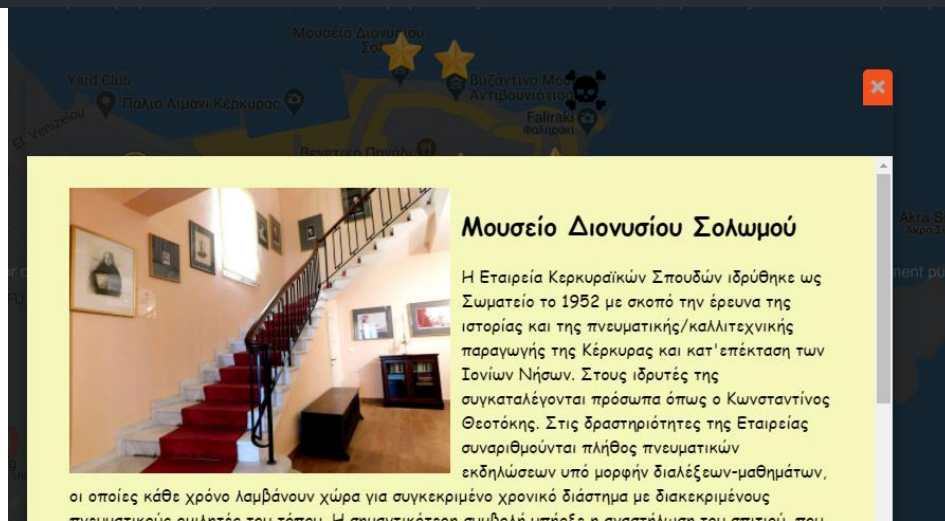
Το φόντου του και το περιεχόμενο του, τα κείμενα,

```
.modal-content {  
  margin: 10% auto;  
  width: 60%;  
  box-shadow: 0 5px 8px 0 rgba(0, 0, 0, 0.2), 0 7px 20px 0 rgba(0, 0, 0, 0.17);  
}
```

```

animation-name: modalopen; /*modal animation*/
animation-duration: var(--modal-duration);
overflow-y: auto; /*modal scroll*/
font-family: Bradley Hand, cursive;
font-size: 100%; }
.modal-content-info {
  /*platos*/
  max-width: 5000px;
  width: 90%;
  margin: 10% auto;
  /*height of information inside the modal*/
  height: 60%;
  max-height: 300vh;
  box-shadow: 0 5px 8px 0 rgba(0, 0, 0, 0.2), 0 7px 20px 0 rgba(0, 0, 0, 0.17);
  animation-name: modalopen; /*modal animation*/
  animation-duration: var(--modal-duration);
  position: relative;
  overflow-y: scroll; /*modal content to scroll*/}
.modal-body {
  padding: 30px 40px;
  background: #f6f7ba;
  height: 300px;
}
/*                                MODAL BODY BACKGROUND FOR MAIN MENU                                */
.modal-bodyyss {
  padding: 30px 40px;
  /* background: #fff; */
  background: #fffff9;
  height: 200px;
}

```



Το κουμπί κλεισίματος(πάνω αριστερά) και την κίνηση του.

```
.close {
  color: #aaa;
  background: #f4511e;
  border-top-left-radius: 5px;
  border-top-right-radius: 5px;
  border: 2px;
  float: right;
  font-size: 28px;
  font-weight: bold; }
.close:hover,
.close:focus {
  color: #aaa;
  color: #000;
  background: #f4511e;
  border-top-left-radius: 6px;
  border-top-right-radius: 6px;
  border-color: brown;
  border-style: dotted;
  text-decoration: none;
  cursor: pointer; }
```

Για να έχουμε modal με εικόνα στο κείμενο,

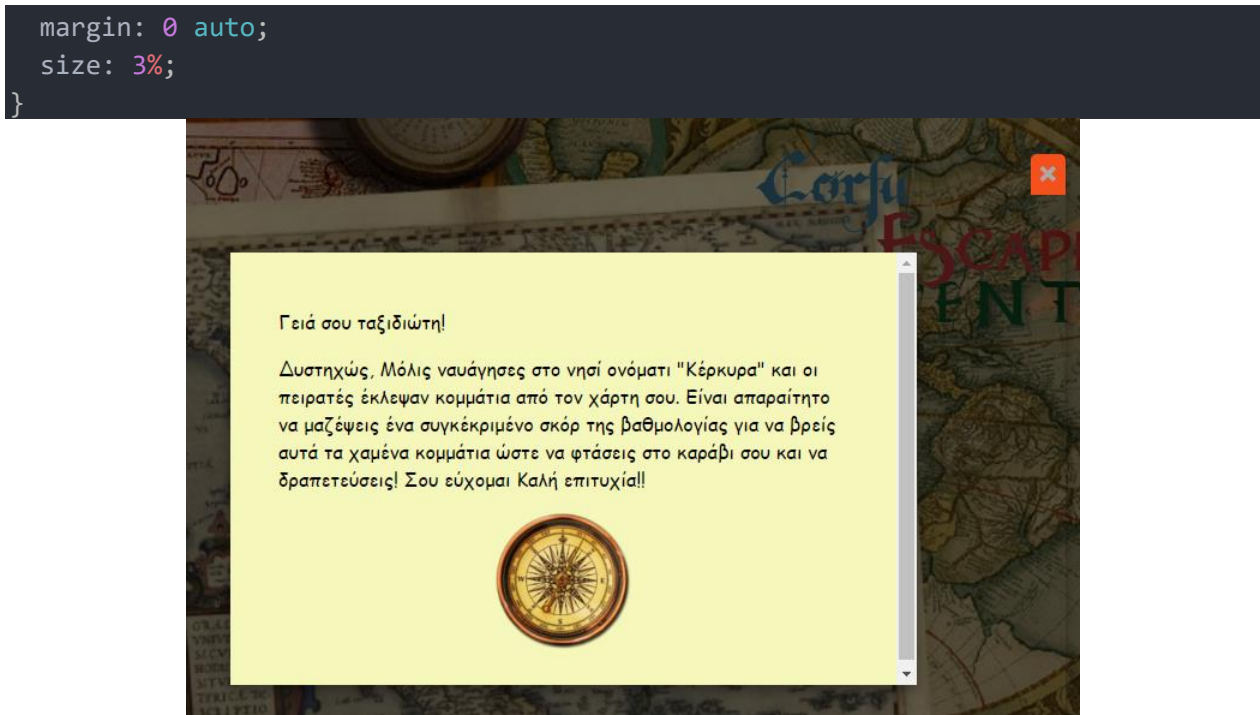
```
.ImageInfo {
  max-width: 50%;
  height: auto;
  /* position relative = > The element is positioned relative to its normal
  position, so "left:20px" adds 20 pixels to the element's LEFT position*/
  position: relative;
  float: left;
  margin-right: 10px;
  margin-bottom: 5px; }
```

Για την κίνηση όλων των modals.

```
@keyframes modalOpen {
  from {
    opacity: 0;
  }
  to {
    opacity: 1;
  }
}
```

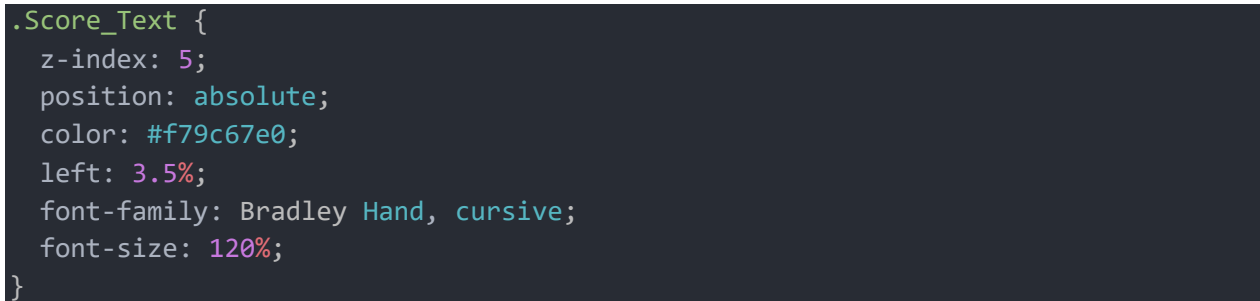
Επίσης, ξεχωριστά έγινε και η δήλωση της πυξίδας του modal , “Story”, του μενού.

```
.compass {
  position: relative; /*move when display changes*/
  /*stay in the center*/
  display: block;
```

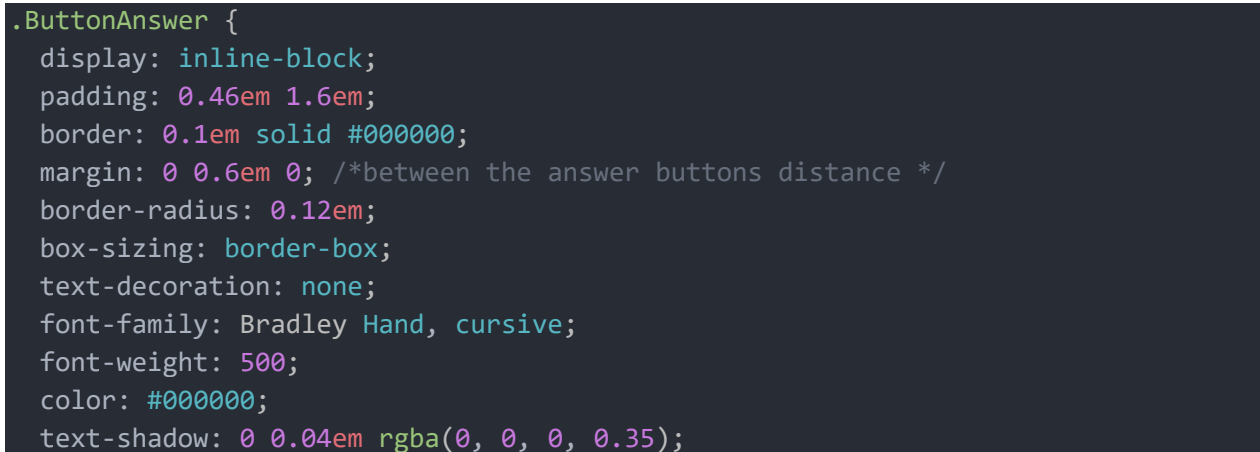


Και αντίστοιχα σε μία άλλη περίπτωση ένας σκελετός.

Για να εμφανίζεται το σκορ του παιχνιδιού,



Για τα κουμπιά των κουίζ, το χρώματα τους, την κίνηση τους όταν τα πατάει ο χρήστης και αλλαγή χρώματος.



```

background-color: #10816e91;
text-align: center;
font-size: 95%;
transition: all 0.15s;
}
.ButtonAnswer:hover {
text-shadow: 0 0 2em rgba(255, 255, 255, 1);
color: #14201b;
border-color: #192720;
border-width: 0.2em;
}
.ButtonAnswer:focus {
background-color: #8f1010a8; /*when wrong answer clicked, show red color*/ }

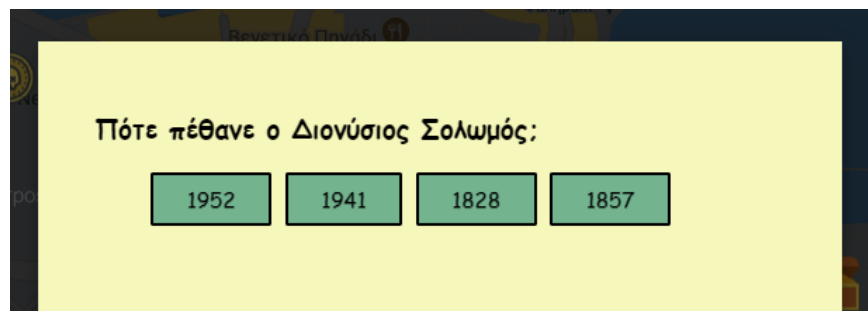
```

Όμως υπάρχει και μία μεταβλητή με τον ίδιο ακριβώς κώδικα , αλλά η διαφορά είναι στο χρώμα του focus η “ButtonAnswer1”.

```

.ButtonAnswer1:focus {
background-color: #27ad15a8; }

```



Τέλος, για το κουμπί που επιστρέφει στην αρχική σελίδα από τον χάρτη,

```

.HomePage {
display: table-cell;
position: relative; /*it doesnt move to the page*/
left: 45%;
bottom: 30%;
width: 100px;
height: 50px;
font-size: 16px; /*Button Size*/
text-align: center;
text-decoration: none;
color: #f5f5f5; /*text color*/
background-color: #2e8b57;
border-radius: 12px; /*property to add rounded corners to a button*/
border: #f5deb3; /*border color*/
}

```

```
transition-duration: 0.4s;
cursor: pointer;
box-shadow: 0 8px 16px 0 rgba(0, 0, 0, 0.2), 0 6px 20px 0 rgba(0, 0, 0, 0.19);}
```

Το επόμενο αρχείο της δομής μας είναι το “App.js”

```
└─ src/
   ├── App.css
   ├── App.js
   ├── Index.js
   └── Index.css
```

Σε αυτό το javascript αρχείο βρίσκεται το αρχικό μενού, το κυρίως μας πρόγραμμα.

Για να λειτουργούν τα στοιχεία αρμονικά μεταξύ τους εισάγουμε πρώτα τις απαραίτητες βιβλιοθήκες, τις βιβλιοθήκες του “react”, και για τα components, το ReactDOM (για να ενώσει τα αντικείμενα/components), το “react-bootstrap” (βιβλιοθήκη των modals), το αρχείο Map από τον φάκελο “components” και το css αρχείο που μόλις δημιουργήσαμε.

```
import React, { Component } from "react";
import ReactDOM from "react-dom";
import { Modal } from "react-bootstrap";
import Map from "./components/Map";
import "./App.css";
```

Έπειτα εισάγουμε όλες τις εικόνες που χρησιμοποιούμε στην αρχική σελίδα, δηλαδή το φόντο, το λόγγο και την πυξίδα.

```
import Logo from "./assets/HomePage/Logo.png";
import menu_map from "./assets/HomePage/Pirate-Wallpaper1.jpg";
import compass from "./assets/HomePage/compass.png";
```

Αμέσως μετά την εισαγωγή είναι η κλάση μας, Corfu την οποία την κάνουμε και εξαγωγή. Σε αυτή την κλάση υπάρχει ένας constructor-κατασκευαστής. Είναι μια μέθοδος που καλείται αυτόματα κατά τη δημιουργία ενός αντικειμένου από μια τάξη. Μπορεί να χειριστεί τα αρχικά στοιχεία ρύθμισής όπως την προεπιλογή ορισμένων ιδιοτήτων του αντικειμένου ή τη λογική να ελέγξουμε τα επιχειρήματα που πέρασαν. Με απλά λόγια, ο κατασκευαστής βοηθά στην κατασκευή πραγμάτων.

Τα Props είναι ορίσματα που μεταφέρονται σε στοιχεία React. Μεταβιβάζονται σε στοιχεία μέσω χαρακτηριστικών HTML. Το React Props είναι σαν ορίσματα συνάρτησης σε JavaScript και χαρακτηριστικά σε HTML. Είναι μόνο για ανάγνωση, είτε δηλωθεί ένα στοιχείο ως συνάρτηση είτε ως κλάση, δεν πρέπει ποτέ να τροποποιεί τα δικά του στηρίγματα.

Τα στοιχεία React έχουν ένα ενσωματωμένο αντικείμενο κατάστασης (state). Το state είναι όπου αποθηκεύετε τιμές ιδιοτήτων που ανήκουν στο στοιχείο, όταν το αντικείμενο κατάστασης αλλάζει, το στοιχείο εκδίδεται ξανά. Σε αυτό δηλώνουμε τις μεταβλητές εμφάνισης των modals και του χάρτη ως ψευδής για να μην εμφανίζονται συνεχώς πάνω στην αρχική σελίδα αλλά μόνο όταν επιθυμούμε εμείς.

```
export default class Corfu extends Component {
  constructor(props) {
    super(props);
    this.state = {
      showcomponent: false,
```

```

    story: false,
    credits: false
  };

  this.showmap = this.showmap.bind(this);
  this.showstory = this.showstory.bind(this);
  this.showcredits = this.showcredits.bind(this);
}

```

Με το `this` και το `bind`, την δέσμευση είναι απαραίτητη για να λειτουργήσει «αυτό» στην επιστροφή κλήσης, να αντιπροσωπεύει το στοιχείο που κατέχει τη μέθοδο.

Άρα οι συναρτήσεις που τις δεσμεύσαμε για την αλλαγή της εμφάνισης των `modal` και το άνοιγμα του χάρτη γίνεται,

```

showmap() {
  this.setState(prevState => ({
    showcomponent: !prevState.showcomponent
  }));
}

```

Στην ουσία αλλάζουμε την τιμή της μεταβλητής από ψευδής σε αληθής, να μην είναι η προηγούμενη τιμή του. Με την `setState` καθίσταται δυνατή αυτή η αλλαγή, προγραμματίζει μια ενημέρωση στο αντικείμενο κατάστασης ενός στοιχείου. Όταν αλλάζει η κατάσταση, το στοιχείο αποκρίνεται με εκ νέου απόδοση.

Αντίστοιχα είναι και οι άλλες δύο συναρτήσεις για τις μεταβλητές των `modals` και πάμε στην μέθοδο `render`. Αναφέρεται σε μια τεχνική για κοινή χρήση κώδικα μεταξύ των στοιχείων `React` χρησιμοποιώντας ένα στήριγμα του οποίου η τιμή είναι μια συνάρτηση και το αποτέλεσμα το επιστρέφει στην οθόνη.

```

render() {
  if (this.state.showcomponent) {
    return (
      <div className="HOL0">
        <Map />
      </div>
    );
  }
  return (

```

Πριν κάνουμε όμως την επιστροφή λέμε στο πρόγραμμα να ελέγξει εάν έχει πατηθεί το κουμπί “play”, τότε να επιστρέψει το αντικείμενο “Map”, τον χάρτη. Εάν αυτό δεν ισχύει τότε συνεχίζει και εμφανίζει το φόντο του μενού, το λόγκο, τα τρία κουμπιά “play”, “story” και “credits”. Όλα αυτά τα βάζουμε μέσα σε ένα `div` για ευκολία δικιά μας και για εύκολη οργάνωση και επεξεργασία των στοιχείων της διεπαφής χρήστη. Στο κουμπί έχουμε ένα συμβάν `onClick`, το οποίο εμφανίζεται όταν ο χρήστης κάνει κλικ πάνω του. Αυτό το συμβάν καλεί την συνάρτηση `showmap()`, που φτιάξαμε για να εμφανιστεί ο χάρτης, αλλά πριν εκτελεστεί αυτό το γεγονός έχει γίνει ο έλεγχος `if`.

```

<div className="App">
  <img className="Logo" src={Logo} alt="" />
  <img className="Menu" src={menu_map} alt="" />

```

```

    { /* /* play button , it opens the map */ }
    <button className="Play_button" onClick={this.showmap}>
      Play
    </button>

```

Αμέσως μετά το κουμπί αυτό έχουμε τα modals, σε ένα εμφωλευμένο div.

Έχουμε το κουμπί “Story”, που μόλις πατηθεί εμφανίζεται το modal, όπου το έχουμε ονομάσει για την εύκολη τροποποίηση του στο css όπως και σε όλες τις μεταβλητές του κώδικα. Στο Modal λέμε να εμφανίζεται και να κρύβεται σύμφωνα με τις αρχικές τιμές που έχουν δηλωθεί στον κατασκευαστή και έπειτα στην επικεφαλίδα του να υπάρχει μόνο το κουμπί κλεισίματος.

```

<div className="Story">
  <button className="Story_button" onClick={this.showstory}>
    Story
  </button>
  <Modal
    className="modal"
    show={this.state.story}
    onHide={this.showstory}
  >
    <Modal.Header className="modal-header">
      { " "}
      <button className="close" onClick={this.showstory}>
        <span aria-hidden="true">&times;</span>
      </button>
    </Modal.Header>
    <Modal.Body className="modal-content">
      <p>Γειά σου ταξιδιώτη!</p>
      <p>
        Δυστυχώς, . . .
      </p>
      <img className="compass" src={compass} alt="show compass"></img>
    </Modal.Body>
  <Modal.Footer></Modal.Footer>
</Modal>
</div>

```

Στο σώμα του (“Modal.Body”) εμφανίζουμε μία παράγραφο τις ιστορίας του παιχνιδιού και αντίστοιχα στο credits τα στοιχεία του κατασκευαστή του παιχνιδιού και ότι πρόκειται για πτυχιακή εργασία. Εκτός του κειμένου εμφανίζεται και μία εικόνα, στην συγκεκριμένη περίπτωση μια πυξίδα.

Ακριβώς κάτω από αυτό το div υπάρχει το div του credits και μετά κλείνει το αρχικό μας div “App”.

Στο τέλος υπάρχει μια γραμμή που λέει το εξής, για να αποδώσουμε ένα στοιχείο React(την κλάση μας) σε έναν ριζικό κόμβο DOM, περνάμε και τα δύο στο ReactDOM.render (). Το ονομάζουμε "ριζικό" κόμβο DOM, διότι τα πάντα μέσα σε αυτό θα διαχειρίζεται η React DOM και είναι η μέθοδος Document.getElementById () που επιστρέφει ένα στοιχείο Element που αντιπροσωπεύει το στοιχείο του οποίου η ιδιότητα id ταιριάζει με την καθορισμένη συμβολοσειρά.

```

    </div>
  );
}
}
ReactDOM.render(<Corfu />, document.getElementById("root"));

```

Τα επόμενα πρόγραμμα, index.js και index.css.

```

├── App.css
├── App.js
├── Index.js
├── Index.css
├── mapStyles.js
└── serviceWorker.js

```

Στο πρώτο έγινε μία μικρή αλλαγή, έγινε εισαγωγή της βιβλιοθήκης App ώστε η εφαρμογή να επιστρέφει των κώδικα του αρχικού μενού (App.js).

```

import React from "react";
import ReactDOM from "react-dom";
import "./index.css";
import App from "./App";
import * as serviceWorker from "./serviceWorker";
ReactDOM.render(<App />, document.getElementById("root"));

```

Το δεύτερο αρχείο, το index.css δεν το πειράζουμε και το serviceWorker.js όπου και αναφέρθηκε στην αρχή τι κάνει και έτσι έχουμε το mapStyles.js.

Πρόκειται για ένα μεγάλο αρχείο και δεν θα βάλουμε όλο τον κώδικα σε αυτή την εργασία. Αυτό το αρχείο είναι γραμμένο σε Javascript και αλλάζει τον τρόπο εμφάνισης του χάρτη μας, διότι δεν θέλαμε να είναι ο καθορισμένος της Google. Επιθυμούσαμε ένα στυλ πιο κοντά σε πειρατικό χάρτη και κατόπιν έρευνας βρέθηκε το επιθυμητό στυλ από την πλατφόρμα, snazzymaps.com. Για να λειτουργήσει όμως θα εισαχθεί στον κώδικα που καλούμε τον χάρτη.

```

├── src/
│   ├── App.css
│   ├── App.js
│   ├── Index.js
│   ├── Index.css
│   ├── mapStyles.js
│   ├── serviceWorker.js
│   ├── Components
│   │   ├── Modal
│   │   │   └── Info.json
│   │   ├── Quiz
│   │   │   └── quizdata
│   │   │       └── Questions.js
│   └── Map.js
└── assets
    ├── HomePage
    ├── Infoimg
    └── Some files .svg

```

Βρισκόμαστε στον πρώτο φάκελο, το “Components”. Σε αυτόν υπάρχει ένα αρχείο με την ονομασία “Map.js”, που είναι ο χάρτης μας, και άλλοι δύο φάκελοι που περιέχουν τα json αρχεία με τις πληροφορίες των Quiz και τις ερωτήσεις τους.

Στον φάκελο «Modal», έχουμε το αρχείο “info.js”. Είναι ένα αρχείο JSON όπου όλες οι πληροφορίες των σημείων του χάρτη είναι αποθηκευμένες σε έναν πίνακα. Τις αποθηκεύσαμε σε έναν πίνακα για να γίνεται εύκολα το κάλεσμά του στον κώδικά μας αλλά και γιατί έτσι είναι η σωστή σύνταξη ενός αρχείου με κατάληξη .json. Αυτόν τον πίνακα του δώσαμε την ονομασία “museumInfo”, και περιέχει τέσσερα αντικείμενα. Το κάθε αντικείμενο έχει ταυτότητα (id), τίτλο και ένα εμφωλευμένο πίνακα (information) με ένα αντικείμενο. Το αντικείμενο αυτού του πίνακα είναι ένα και είναι μία παράγραφος με πληροφορίες του κάθε μουσείου ή σημείου του χάρτη. Τα λη τοποθετήθηκαν για να γίνεται διαχωρισμός των προτάσεων, αλλαγή σειράς.

```
{
  "museumInfo": [
    {
      "id": "1",
      "title": "Σινοϊαπωνικό Μουσείο",
      "information": [
        "Στα Ανάκτορα των Αρχαγγέλου Μιχαήλ και Αγίου Γεωργίου στεγάζεται το Μουσείο Ασιατικής Τέχνης της Κέρκυρας, . . .
      ]
    }, . . .
  ]
}
```

Στον άλλο φάκελο, «Quiz», είναι οι ερωτήσεις. Το αρχείο έχει την ονομασία “Questions.js” και διαφορετική κατάληξη από το προηγούμενο, αλλά είναι πάλι αρχείο JSON. Η διαφορά είναι ότι έχει γίνει διαφορετική δήλωση του πίνακα.

Δηλαδή, ο πίνακας “Questions” δηλώνεται στην αρχή σαν μεταβλητή και περιέχει πέντε αντικείμενα, όσες και οι ερωτήσεις. Το κάθε αντικείμενο έχει την ερώτηση(question), αριθμό (questionId), έναν εμφωλευμένο πίνακα(answers) με τις πιθανές απαντήσεις και την σωστή απάντηση (correct), που θα γίνεται ο έλεγχός της σε άλλο πρόγραμμα.

```
var Questions = [
  {
    question: "Πότε πέθανε ο Διονύσιος Σολωμός;",
    quesstionId: "1",
    answers: ["1952", "1941", "1828", "1857"],
    correct: " "
  },
]
export default Questions;
```

Τον πίνακα τον κάνουμε στο τέλος εξαγωγή σαν επέκταση αρχείου json.

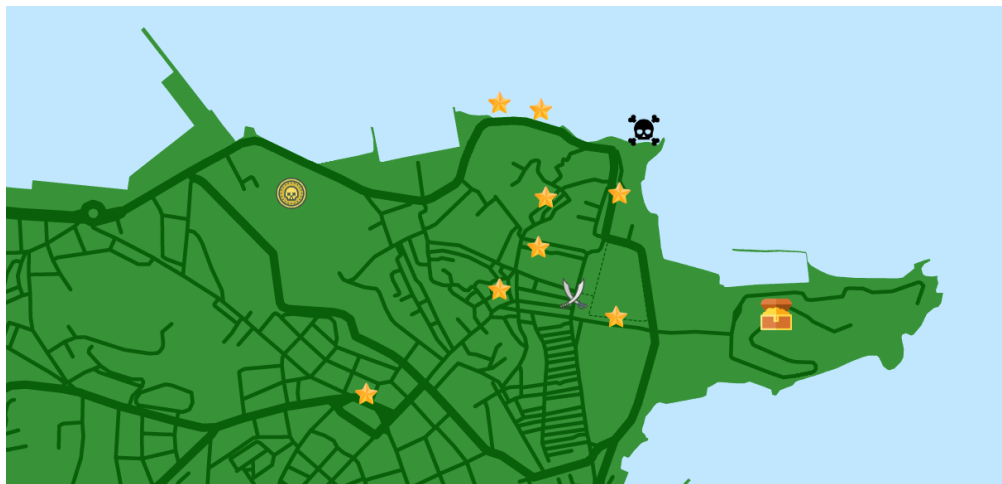
Το δεύτερο κυρίως πρόγραμμα του παιχνιδιού μας (“Map.js”) ανοίγει κατόπιν του πατήματος του κουμπιού “play” στο αρχικό μενού.

Σε αυτό των κώδικα εισάγουμε πρώτα τις απαραίτητες βιβλιοθήκες.

```
import React, { Component } from "react";
import { Map, GoogleApiWrapper, Marker } from "google-maps-react";
import { Modal } from "react-bootstrap";
import Confetti from "react-confetti";
import mapStyles from "../mapStyles";
import Questions from "../Quiz/quizdata/Questions";
import App from "../App";
```

Εισάγουμε τις βιβλιοθήκες του React καθώς και τους components ώστε να μπορούμε να τους χρησιμοποιήσουμε παρακάτω. Απαραίτητη βιβλιοθήκη είναι η “google-maps-react” διότι μέσα από αυτή μπορούμε να χρησιμοποιήσουμε τα εργαλεία του χάρτη της Google αλλά και να τον διαχειριστούμε. Μετά είναι η “react-bootstrap” για την χρήση των Modals και η “react-confetti” για τα κομφετί.

Για να μην εμφανίζεται ο χάρτης μας με την μορφοποίηση της Google και τις επιλογές της (δορυφορική οπτική, χάρτης), όπως αναφέρθηκε ήδη παραπάνω εισάγουμε το αρχείο mapStyles.js που δίνει την επιθυμητή μορφή ενός πειρατικού χάρτη.



Εισάγουμε το αρχείο «Questions.js», τον πίνακα με τις ερωτήσεις και το πρόγραμμα «App», θα αναφερθεί αργότερα το γιατί. Επίσης εισάγουμε το άλλο αρχείο JSON, με τις πληροφορίες των Modal «info.json» καθώς και όλες τις εικόνες που θα χρησιμοποιηθούν στην εφαρμογή.

```
import { museumInfo } from "../Modal/info.json";
import mouseio from "../assets/infoimg/prosopsi-mouseiou_Image.jpg";
import trapeza from "../assets/infoimg/moneymuseum.png";
import solomo from "../assets/infoimg/solomo.jpg";
import anoutsiaata from "../assets/infoimg/annunziata-corfu-2.jpg";
import skeleton from "../assets/infoimg/skeleton.png";
```

Εισάγουμε και τα εικονίδια των σημείων του χάρτη, τα αστέρια κλπ σε μορφή Svg (θα γίνει εξήγηση των αρχείων στο τέλος).

```
var markericon = require("../assets/star.svg");
var markerfinal = require("../assets/chest.svg");
var markerbonus = require("../assets/coin.svg");
var markerfight = require("../assets/saber.svg");
var markergameover = require("../assets/risk-skull.svg");
```

Οπότε τώρα φτιάχνουμε μια κλάση με την ονομασία “MapContainer” που επεκτείνει τα αντικείμενα του προγράμματος. Όπως και στον κώδικα “App”, την αρχική μας σελίδα, έχουμε την μέθοδο constructor (κατασκευαστή) για να δημιουργήσουμε τα αντικείμενα της κλάσης, να αρχικοποιήσουμε τις ιδιότητες τους. Τα props, είναι σαν ορίσματα συνάρτησης και τα στέλνουμε στο στοιχείο ως χαρακτηριστικά. Επίσης είναι ο τρόπος με τον οποίο μεταδίδουμε δεδομένα από το ένα στοιχείο στο άλλο, ως παράμετροι.

Τα στοιχεία React έχουν ενσωματωμένο αντικείμενο κατάστασης (built-in state object).

```
export class MapContainer extends Component {
  constructor(props) {
    super(props);
    this.state = {
```

Στο αντικείμενο κατάστασης (state), αποθηκεύουμε τις αρχικές τιμές αυτών των εξής αντικειμένων,

Ένας πίνακας stores, μεταβλητές εμφάνισης των modals πληροφορίας(από 1- 5) και των ερωτήσεων (εμφανιζόμενων ή μη, 6-13) καθώς και έναν πίνακα που αποθηκεύονται οι ερωτήσεις από το json αρχείο(15), η score μεταβλητή για να κρατάμε τους πόντους που μαζεύει ο παίκτης κατά την διάρκεια του παιχνιδιού και η isActive για τον έλεγχο των απαντήσεων του κουίζ.

```
1. show: false,
2. showasia: false,
3. showsolomo: false,
4. showmoney: false,
5. showmuseum: false,
  //Game initilizations
6. showq: false,
7. showq1: false,
8. showq2: false,
9. showq3: false,
  // Test Score
10. testscore: false,
11. testscore1: false,
    activeMarker: {},
  //Final Result
12. finale: false,
13. finale1: false,
14. HomePage: false,
//QUIZ
15. questions: [], //array of questions from json file,
    score: 0, //Games Score
    isActive: false
```

Η activeMarker χρειάζεται γιατί όταν ο χρήστης θα πατάει πάνω στο σημείο του χάρτη θα ανοίγει το Modal του σημείου αυτού. Θα το δούμε παρακάτω στον κώδικα.

Επίσης μέσα στην state, έχουμε δηλώσει έναν πίνακα (stores) με τα σημεία του χάρτη,

```
stores: [ { name: "a",
  latitude: 39.625769,
```

```
longitude: 19.924659,
marker: markericon, //star svg
scaledSize: new window.google.maps.Size(17, 17),
onclick: this.showasiastory      }, { . . . }]
```

Όπου μέσα σε κάθε αντικείμενο του πίνακα αυτού έχουμε μία ονομασία του marker , τις συντεταγμένες (latitude/γεωγραφικό πλάτος και longitude/γεωγραφικό μήκος), την εικόνα που θα έχει (svg συγκεκριμένα), το μέγεθος του (πόσο μεγάλο θα είναι στην οθόνη) και τέλος μία onclick συνάρτηση. Με αυτή την συνάρτηση «λέμε» στο πρόγραμμα , ότι όταν πατήσει ο χρήστης πάνω στο σημείο αυτό κάλεσε την συνάρτηση με αυτό το όνομα (this.showasiastory).

Σε κάθε όμως αντικείμενο στην μέθοδο onclick (κίτρινο χρώμα), καλούμε διαφορετική συνάρτηση για το κάθε σημείο. Δηλαδή για τα σημεία που έχουν μηνύματα με πληροφορίες,

1. Σινοιαπωνικό Μουσείο Κέρκυρας,

```
{
    name: "a",
    latitude: 39.625769,
    longitude: 19.924659,
    marker: markericon, //star svg
    onclick: this.showasiastory //onclick => show info },
```

2. Μουσείο Διονύσιου Σολωμού

```
{
    name: "b",
    latitude: 39.627578,
    longitude: 19.92153,
    marker: markericon, //star svg
    onclick: this.showsolomostory //onclick => show info },
```

3. Μουσείο Χαρτονομισμάτων

```
{
    name: "c",
    latitude: 39.625695,
    longitude: 19.922731,
    marker: markericon, //star svg
    onclick: this.showmoneystory      },
```

4. Μουσείο Ανουτσιάτα

```
{
    name: "h",
    latitude: 39.623858,
    longitude: 19.921549,
    marker: markericon, //star svg
    onclick: this.showmuseumstory //onclick => show info      },
```

Για τα σημεία που είναι οι ερωτήσεις αντίστοιχα,

1. Βυζαντινό Μουσείο Αντιβουινιώτισσας

```
{
    name: "d",
    latitude: 39.627437,
```

```
longitude: 19.922615,
marker: markericon, //star svg
onclick: this.showquiz //onclick => show quiz },
```

2. Σπιανάδα

```
{
  name: "f",
  latitude: 39.623315,
  longitude: 19.924562,
  marker: markericon, //star svg
  onclick: this.showquiz1 //onclick => show quiz },
```

3. Σαν ρόκο

```
{
  name: "i",
  latitude: 39.621769,
  longitude: 19.918078,
  marker: markericon, //star svg
  onclick: this.showquiz2 //onclick => show quiz },
```

4. Ιόνιος τράπεζα

```
{
  name: "j",
  latitude: 39.624701,
  longitude: 19.922549,
  marker: markericon, //star svg
  onclick: this.showquiz3 //onclick => show quiz },
```

Το επόμενο σημείο είναι το Πεντοφάναρο, όπου καλούμε την “testScoreBattle” συνάρτηση, που θα πούμε παρακάτω τι κάνει.

```
{
  name: "g",
  latitude: 39.623673,
  longitude: 19.923442,
  marker: markerfight, //pirate swords svg
  onclick: this.testScoreBattle //onclick => show game over if score not
  enough },
```

Αντίστοιχα στο Νέο φρούριο και

```
{
  name: "k",
  latitude: 39.62567,
  longitude: 19.916142,
  marker: markerbonus,
  onclick: this.increaseScore //onclick => increase score +10 , },
```

στο Φαλιράκι ,

```
{
  name: "l",
  latitude: 39.626924,
  longitude: 19.92528,
  marker: markergameover, //death svg
  onclick: this.decreaseScore },
```

Και στον τελικό προορισμό, το Παλαιό Ενετικό Φρούριο.

```
{
  name: "e",
  latitude: 39.623256,
  longitude: 19.928705,
  marker: markerfinal, //svg chest
  onclick: this.finaldestination }

```

Ακριβώς μετά το state ,αλλά ακόμα εξακολουθούμε να είμαστε μέσα στον κατασκευαστή, έχουμε τις δηλώσεις των συναρτήσεων μας. Έχουμε συναρτήσεις ξεχωριστές για τα modal (πληροφορίες και εικόνες) , για τις ερωτήσεις (Τα quiz) , για τον έλεγχο του σκορ, για την μείωση του σκορ, την αύξηση του και μία συνάρτηση για την επιστροφή του χρήστη στην αρχική οθόνη.

```
};
//Modal info init
this.showasiastory = this.showasiastory.bind(this);
this.showsolomostory = this.showsolomostory.bind(this);
this.showmoneystory = this.showmoneystory.bind(this);
this.showmuseumstory = this.showmuseumstory.bind(this);
//Quiz init
this.showquiz = this.showquiz.bind(this);
this.showquiz1 = this.showquiz1.bind(this);
this.showquiz2 = this.showquiz2.bind(this);
this.showquiz3 = this.showquiz3.bind(this);
//Test Score
this.testScoreBattle = this.testScoreBattle.bind(this);
//Go to Home Page
this.Go = this.Go.bind(this);
//Increase Score
this.increaseScore = this.increaseScore.bind(this);
//Decrease Score
this.decreaseScore = this.decreaseScore.bind(this); }
```

Στην React η δήλωση των συναρτήσεων γίνεται με το this και το bind και αυτό είναι παρμένο από την Javascript. Η μέθοδος bind () δημιουργεί μια νέα συνάρτηση που, όταν καλείται, ορίζει αυτήν τη λέξη-κλειδί στην παρεχόμενη τιμή, με μια δεδομένη ακολουθία ορισμάτων που προηγούνται οποιωνδήποτε παρεχόμενων κατά την κλήση της νέας συνάρτησης. Βοηθάει πολύ για να ελέγχουμε τα στοιχεία και τους χειριστές συμβάντων.

Κάπως έτσι κλείνει ο κατασκευαστής μας και συνεχίζουμε με τις συναρτήσεις και τις αλλαγές που επιφέρουν στον κώδικα μας.

Η πρώτη συνάρτηση που συναντάμε είναι η “ Display Markers ”. Αυτή την συνάρτηση την καλούμε με () => διότι μεταφέρουμε ένα πρόγραμμα χειρισμού συμβάντων (όπως onClick) σε ένα στοιχείο. Η χρήση μιας συνάρτησης βέλους στην απόδοση δημιουργεί μια νέα συνάρτηση κάθε φορά που το στοιχείο αποδίδεται. Έτσι επιστρέφουμε, από τον πίνακα του κατασκευαστή, με την μέθοδο map των πίνακα με τους δείκτες του χάρτη, όπου για κάθε δείκτη, επιστρέφεται ένα στοιχείο ως αντικείμενο Marker του Google maps. Μέσα στο <Marker> επιστρέφεται αυτά που του έχουμε ορίσει, η ταυτότητα του, η γεωγραφική του θέση, το εικονίδιο που θα έχει (το μέγεθος του και την κίνηση του) καθώς και την onclick συνάρτηση του.

```
displayMarkers = () => {
```

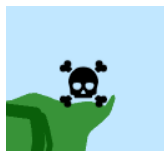
```

return this.state.stores.map((store, index) => {
  return (
    <Marker
      key={index}
      id={index}
      position={{
        lat: store.latitude,
        lng: store.longitude
      }}
      icon={{
        url: store.marker,
        scaledSize: new window.google.maps.Size(38, 38)
      }}
      animation={window.google.maps.Animation.BOUNCE}
      onClick={store.onclick}
    />
  );
});
};

```

Αμέσως μετά είναι οι συναρτήσεις που συσχετίζονται με το score. Έχουμε πέντε περιπτώσεις, άρα και πέντε συναρτήσεις.

Οι πρώτες δύο είναι οι συναρτήσεις δύο αντίθετων λογικών. Για την πρώτη, λέμε με την setState, να ενημερωθεί η μεταβλητή score και να αυξηθεί συν δέκα βαθμούς. Δηλαδή, όταν πατηθεί στον χάρτη το συγκεκριμένο σημείο (νόμισμα) λειτουργεί σαν επίπεδο μπόνους, και ο παίκτης κάθε φορά που θα το πατάει θα κερδίζει δέκα πόντους. Ενώ στην δεύτερη συνάρτηση, κάθε φορά που ο χρήστης θα πατάει το σημείο (νεκροκεφαλή) θα χάνει όλο το σκορ που έχει μαζέψει και θα μηδενίζεται.



```

increaseScore = () => {
  if (!this.state.workoncxtrapoints) {
    this.setState((state, prevState) => ({
      score: state.score + 10,
      workoncxtrapoints: !prevState.workoncxtrapoints
    }));
  }
};

decreaseScore = () => {
  this.setState(state => ({
    score: (state.score = 0)
  }));
};

```

Η Τρίτη περίπτωση συναρτήσεων είναι η “Score”, που δέχεται ως όρισμα ένα index και μία μεταβλητή isActive, τις απαντήσεις από το κλικ που έκανε ο χρήστης και ελέγχει εάν η απάντηση ταιριάζει με την σωστή απάντηση. Εάν είναι η σωστή απάντηση ο παίκτης κερδίζει δέκα πόντους και σε κάποιες περιπτώσεις είκοσι. Εάν δεν είναι η απάντηση καμία από όλες τις περιπτώσεις τότε δεν γίνεται τίποτα (προσωπική επιλογή). Με την άλλη μεταβλητή αλλάζουμε την τιμή της, ώστε όταν πατιέται η σωστή απάντηση θα καλεί τον σωστό κώδικα στο css αρχείο για να αλλάζει το χρώμα του κουμπιού από κόκκινο σε πράσινο και το αντίστροφο. Αυτή την διαδικασία την κάνει η prevState , για να μην χρειάζεται να αλλάζει ο προγραμματιστής την τιμή συνέχεια από true σε false κλπ. Διότι αν γίνεται πολλές φορές η αλλαγή αυτή δεν είναι δυνατόν να θυμάται σε ποιο σημείο πρέπει να τις κάνει αυτές τις αλλαγές.

```
Score = (index, isActive) => {
  if (index === "1857") {
    this.setState(prevState => ({
      score: this.state.score + 10,
      isActive: !prevState.isActive
    })); . . . }
}
```

Η συνάρτηση “testScoreBattle”, ελέγχει αν το σκορ είναι ανάμεσα στο μηδέν και στο τριάντα, τότε εμφανίζεται ένα μήνυμα στον χρήστη και συγκεκριμένα ένα modal , που γράφει ότι ο παίκτης έχει αιχμαλωτισθεί από τους πειρατές και πως πρέπει να απαντήσει και άλλες ερωτήσεις σωστά ώστε να φθάσει πιο κοντά στο τέλος του παιχνιδιού. μπορείτε να δείτε την ερώτηση και να απαντήσετε στο κουίζ. Εάν ο βαθμός που έχει συγκεντρωθεί είναι μεγαλύτερος του τριάντα εμφανίζεται ένα επιπλέον κουίζ που δίνει την δυνατότητα στον παίκτη να κερδίσει περισσότερους βαθμούς.

Οπότε, κάνει πρώτα τον έλεγχο «αν» του σκορ, και λέμε με την setState να αλλάξει η τιμή της μεταβλητής “testScore1” σε διαφορετική από αυτή που είχε, δηλαδή true, και να εμφανιστεί το μήνυμα των πειρατών. Αλλιώς, αντιστοίχως αλλάζει η τιμή της μεταβλητής “testScore” και εμφανίζεται η ερώτηση του κουίζ και καλούμε την προηγούμενη συνάρτηση “Score”. Σε αυτή την συνάρτηση περνάμε σαν παράμετρο την ήδη παράμετρό μας (index) , έτσι ώστε να ελεγχθεί η απάντηση που δόθηκε από τον χρήστη αν είναι σωστή ή όχι.

```
testScoreBattle = (index, isActive) => {
  //If not enough score
  if (this.state.score <= 30 && this.state.score >= 0) {
    this.setState(prevState => ({
      testScore1: !prevState.testScore1
    }));
  } else {
    //show an extra quiz question
    this.setState(prevState => ({
      testScore: !prevState.testScore
    }));
    //check the answer
    this.Score(index, isActive); } };
```

Η Πέμπτη και τελική συνάρτηση που αφορά την βαθμολογία είναι η “finaldestination”. Εδώπερα δεν περνάμε κανένα όρισμα, απλώς δέχεται το σκορ και κάνει ελέγχους. Εάν η βαθμολογία είναι μεγαλύτερη του πενήντα τότε αλλάζουμε την τιμή της μεταβλητής σε true και εμφανίζεται το μήνυμα ολοκλήρωσης του παιχνιδιού. Εάν δεν ισχύει η παραπάνω περίπτωση τότε αλλά η τιμή της μεταβλητής “finale1” και ο

χρήστης βλέπει ένα μήνυμα που το αναφέρει πώς πρέπει να ολοκληρώσει όλες τις ερωτήσεις για να κερδίσει το παιχνίδι.

```
finaldestination = () => {
  if (this.state.score >= 50) {
    this.setState(prevState => ({
      finale: !prevState.finale
    }));
  } else {
    this.setState(prevState => ({
      finale1: !prevState.finale1 })); } };
```

Μετά είναι οι συναρτήσεις των μηνυμάτων που εμφανίζονται στην οθόνη του χρήστη. Δηλαδή τα modal των πληροφοριών και αντίστοιχα των ερωτήσεων. Όπου και στις δύο περιπτώσεις ισχύει αυτό που ήδη αναφέραμε, η αλλαγή της τιμής της εμφάνισης των μηνυμάτων που δηλώθηκαν στον κατασκευαστή.

```
showasiastory = () => {
  this.setState(prevState => ({
    showasia: !prevState.showasia //set the variable to true })); };
```

Για τις ερωτήσεις,

```
showquiz = () => {
  this.setState(prevState => ({
    showq: !prevState.showq //set the variable to true })); };
```

Για την συνάρτηση “Go”, η οποία όπως αναφέραμε, είναι αυτή που αλλάζει την τιμή της μεταβλητής με την χρήση της setState, ώστε να μπορεί ο παίκτης να επιστρέψει στο αρχικό μενού.

```
Go = () => {
  this.setState(prevState => ({
    HomePage: !prevState.HomePage })); };
```

Αφού δηλώσαμε τις βιβλιοθήκες, τις μεταβλητές και τις συναρτήσεις και γράψαμε τις αλλαγές που επιβάλει η κάθε μία από αυτές, έχουμε την μέθοδο “render”.

Στο αρχείο “App.js”, εξηγήσαμε τι κάνει η μέθοδος αυτή ακριβώς, αναφέρεται σε μια τεχνική για κοινή χρήση κώδικα μεταξύ των στοιχείων React χρησιμοποιώντας ένα στήριγμα του οποίου η τιμή είναι μια συνάρτηση και το αποτέλεσμα το επιστρέφει στην οθόνη. Οπότε μέσα σε αυτή την μέθοδο επιστρέφουμε στην οθόνη του χρήστη τον χάρτη. Αλλά πριν από τον div που τον περιέχει έχουμε έναν έλεγχο «αν».

```
render() {
  if (this.state.HomePage) {
    return (
      <div className="HOL0">
        <App />
      </div>
    );
  }
  return (
    <div className="Map"> . . .
```

Σε αυτό το αν ελέγχουμε εάν η μεταβλητή “HomePage” έχει αλλάξει σε true. Αν ισχύει η συνθήκη αυτή τότε «φεύγουμε» από το παιχνίδι, τρέχει ο κώδικας “App.js” και βλέπουμε στην οθόνη μας το αρχικό μενού. Εάν δεν ισχύει η συνθήκη τότε απλώς συνεχίζουμε να βλέπουμε τον χάρτη.

Όλος ο κώδικας μας βρίσκεται μέσα σε ένα div με την ονομασία “Map”. Μέσα σε αυτό το div, είναι το component “Map” ο χάρτης της Google που χρησιμοποιούμε από την βιβλιοθήκη της.

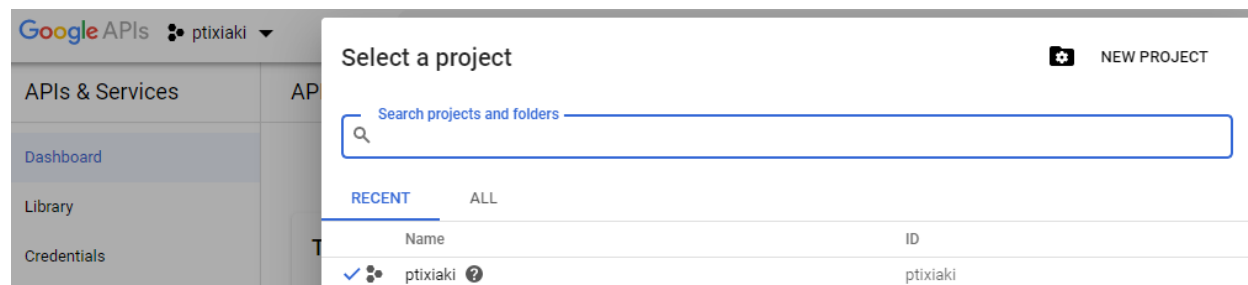
Πριν προχωρήσουμε στο τι κάνει η υπόλοιπη render, πρέπει να αναφέρουμε πως γίνεται το «κάλεσμα» του χάρτη από την “react-google-maps”.

Μετά την κλάση «MapContainer», κάνουμε εξαγωγή του “GoogleApiWrapper”.

```
export class MapContainer extends Component {  
  
  . . . οι συναρτήσεις  
  render() {  
    return (  
      <div className="Map">  
        <Map  
          google={this.props.google}  
          zoom={16} //initial zoom  
          styles={mapStyles} //choose the style of the maps/appearance  
          initialCenter={{ lat: 39.6249838, lng: 19.922346100000027 }}  
        />  
      </div>  
    );  
  }  
}  
export default GoogleApiWrapper({  
  })(MapContainer);
```

Πρόκειται για μια μονάδα Javascript που περισυλλέγει υπηρεσίες και τάξεις για API Google, όπως η υπηρεσία AutocompleteService, Map, Markers και Geocode. Στην περίπτωση μας χρειαζόμαστε τους χάρτες και τα σημεία / δείκτες του, άρα τα google.maps.Map και google.maps.Marker. Οπότε η εντολή αυτή παρέχει διεπαφή αναζήτησης για μέρη και προβλέψεις ερωτημάτων. Μέσα σε αυτή την διαδικασία δίνουμε το Api κλειδί και του λέμε να εξάγει τον χάρτη.

Για να αποκτήσουμε το API κλειδί πρέπει να πάμε στην πλατφόρμα της Google Developers και με τον λογαριασμό μας να φτιάξουμε ένα «έργο» (project).



Από εκεί μετά στη βιβλιοθήκη , ενεργοποιούμε την “Maps JavaScript API” ,



Maps JavaScript API

Google

Maps for your website

MANAGE

✓ API Enabled

Και δημιουργούμε δικό μας κλειδί στην κατηγορία “credentials”. Μόλις πάρουμε το μοναδικό μας κλειδί το συνδέουμε με το πρόγραμμα μας μέσα στην “GoogleApiWrapper”.

```
export default GoogleApiWrapper({  
  apiKey: "AIzaSyA- "  
})(MapContainer);
```

Τώρα πίσω στον υπόλοιπο κώδικα την κλάσης «MapContainer», είμασταν στην render μέθοδο. Όπου εκεί γράφουμε το τί θα επιστραφεί στην οθόνη του υπολογιστή. Αναφέραμε ότι ο χάρτης εισάγεται σαν component , <Map>, και εκεί μέσα δηλώνουμε ότι είναι χάρτης της Google, η μεγέθυνση είναι στο 16%, το στυλ του χάρτη είναι το javascript αρχείο μας “mapStyles” και η αρχική τοποθεσία που θα εμφανίζεται θα είναι η Κέρκυρα.

```
render() {  
  return (  
    <div className="Map">  
      <Map  
        google={this.props.google}  
        zoom={16} //initial zoom  
        styles={mapStyles}  
        initialCenter={{ lat: 39.6249838, lng: 19.922346100000027 }}  
      />  
    </div>  
  );  
}
```

Έπειτα αρχίζουμε και ότι προσθέτουμε θα το εμφανίζουμε πάνω στο χάρτη.

Αρχικά, εμφανίζουμε τα σημεία όλων των αξιοθέατων της πόλης, καλώντας την συνάρτηση “displayMarkers”.

```
initialCenter={{ lat: 39.6249838, lng: 19.922346100000027 }}  
  >  
  {this.displayMarkers()}}
```

Αμέσως μετά χωρίζουμε για ευκολία τα υπόλοιπα στοιχεία σε div, σε κατηγορίες.

Έχουμε τα div, για τις πληροφορίες, για τις ερωτήσεις , για το επίπεδο με την κρυφή ερώτηση και τον τελικό προορισμό.

```

    {this.displayMarkers()}
    {/*          Show the Information inside Modals          */}
    <div className="AsiaModal">...
    <div className="SolomoModal">...
    <div className="MoneyModal">...
    <div className="MuseumModal">...
    {/*          Show the Questions inside Modals          */}
    <div className="GameModal">...
    <div className="GameModal1">...
    <div className="GameModal2">...
    <div className="GameModal3">...
    {/*          Test the Score          */}
    {/*          Version 1 => You get a quiz          */}
    <div className="TestScore">...
    {/*          Version 2 => Deduct points          */}
    <div className="TestScore1">...
    {/*          Final Score          */}
    {/*          Version 1 => You win!          */}
    <div className="FinalDestination">...
    {/*          Version 2 => You need more score points!          */}
    <div className="FinalDestination1">...
  </Map>

```

Στην πρώτη κατηγορία div, σε κάθε τέτοια «κλάση» έχουμε ένα modal για να εμφανίζει τις πληροφορίες.

Έχουμε ονομασία του modal, για να γίνεται εύκολη η οπτική του τροποποίηση μέσω του css αρχείου. Με την εντολή show, δηλώνουμε την εμφάνιση του modal ενώ με την onHide, θα κρύβεται. Εάν δηλαδή η μεταβλητή της onHide είναι αληθής τότε το modal δεν θα εμφανίζεται στην οθόνη και αντίστοιχα για την άλλη εντολή. Με το marker, δηλώνουμε ότι το modal ανήκει σε ένα συγκεκριμένο marker από τον πίνακα που έχουμε δηλώσει, με την size ορίζουμε εάν θα εμφανίζεται σαν μικρό μήνυμα στην οθόνη ή σαν μεγάλο και θα πιάνει όλη την οθόνη.

```

<Modal
  className="hi"
  show={this.state.showasia}
  marker={this.state.activeMarker}
  onHide={this.showasiastory}
  size="lg"
  aria-labelledby="contained-modal-title-vcenter"
>
  <Modal.Header className="modal-header">
    {" "}
    <button className="close" onClick={this.showasiastory}>
      <span aria-hidden="true">&times;</span>
    </button>
  </Modal.Header>
  <Modal.Body className="modal-content-info">
    <img className="ImageInfo" src={mouseio} alt="show museum profile"></img>
    <h2>{museumInfo.map(x => x.title)}</h2>
    <p>{museumInfo.map(x => x.information)}</p>
  </Modal.Body>
  <Modal.Footer></Modal.Footer>
</Modal>;

```

Στο Modal.Header, είναι η «επικεφαλίδα» του μηνυματός μας που περιέχει το κουμπί για να κλείνει. Μόλις ο παίκτης πατήσει πάνω σε αυτό το κουμπί αλλάζει η τιμή της μεταβλητής και κρύβεται το μήνυμά μας.

Στο σώμα του modal, εμφανίζουμε μία εικόνα () και την συνδέουμε με την src με αυτή που επιθυμούμε. Το <h2> θα εμφανίζεται σαν τίτλος του κειμένου, όπου καλούμε τον πίνακα “museumInfo”, από το αρχείο Json που εισήγαμε στην αρχή μαζί με τις βιβλιοθήκες, με την συνάρτηση map.

Η map, λειτουργεί σαν βρόχος πάνω από έναν πίνακα για να αποδώσει το περιεχόμενό του χωρίς να χρειάζεται να γνωρίζουμε το μέγεθος του πίνακα και να ψάχνουμε ένα τα στοιχεία του. Απλώς λέμε κατευθείαν ποιο στοιχείο θέλουμε, πχ. Στο αρχείο json με τις πληροφορίες, έχουμε τα στοιχεία που το καθένα έχει έναν τίτλο, και καλούμε τον συγκεκριμένο να εμφανιστεί.

```
"museumInfo": [
  {
    "id": "1",
    "title": "Σινοϊαπωνικό Μουσείο",
    "information": [
      "Στα Ανάκτορα των Αρ ...." ] . . .
```

Έτσι αντίστοιχα κάνουμε και map σε παράγραφο το περιεχόμενο αυτού του αντικειμένου και εμφανίζεται όλη η πληροφορία.

```
<h2>{museumInfo.map(x => x.title)}</h2>
<p>{museumInfo.map(x => x.information)}</p>
```

Έτσι δηλώνουμε και τα υπόλοιπα σημεία των μουσείων που θέλουμε να έχουν πληροφορίες.



Τώρα, στα modals με τις ερωτήσεις, οι εντολές είναι ίδιες, η μόνη διαφορά είναι στο κυρίως σώμα (“Modal.Body”).

```
<div className="GameModal">
  <Modal
    className="hi"
```

```

    show={this.state.showq}
    marker={this.state.activeMarker}
    onHide={this.showquiz}
  >
  <Modal.Header className="modal-header">
    {" "}
    <button className="close" onClick={this.showquiz}>
      <span aria-hidden="true">&times;</span>
    </button>
  </Modal.Header>
  <Modal.Body className="modal-content">
    <h3>{Questions.map(q => q.question)}</h3>
    <ul>
      {Questions[0].answers.map((answers, index) => (
        <button
          key={index}
          className={this.state.isActive ? "ButtonAnswer1" : "ButtonAnswer"}
          onClick={() => this.Score(answers)}
        >
          {answers}
        </button>
      ))}
    </ul>
  </Modal.Body>
  <Modal.Footer></Modal.Footer>
</Modal>
</div>;

```

Στο σώμα λέμε με την μέθοδο map, να εμφανίσει το περιεχόμενο από το άλλο json αρχείο, “Questions”, δηλαδή τις ερωτήσεις. Αντίστοιχα με πριν, λέμε ποια ερώτηση συγκεκριμένα θα εμφανίζεται στον δείκτη που πατάμε και εμφανίζεται αυτό το μήνυμα.

Την ερώτηση επιλέγουμε να την έχουμε σαν τίτλο, δηλαδή σε μεγάλο μέγεθος, h3.

```

<Modal.Body className="modal-content">
  <h3>{Questions.map(q => q.question)}</h3>

```

Αμέσως μετά λέμε να επιστρέφονται οι απαντήσεις του πίνακα των ερωτήσεων σε μορφή λίστας, με το tag της HTML γλώσσας. Συγκεκριμένα λέμε ότι από τον πίνακα “Questions”, θα επιστραφεί η πρώτη ερώτηση, θέση μηδέν στον πίνακα, θα επιστρέψεις τις απαντήσεις με την μέθοδο map.

```

<ul>{Questions[0].answers.map((answers, index) => ())} </ul>

```

Ο πίνακας,

```

var Questions = [
  {
    //MUSEUM OF SOLOMO
    question: "Πότε πέθανε ο Διονύσιος Σολωμός;",
    quesstionId: "1",
  }
]

```

```
answers: ["1952", "1941", "1828", "1857"],
correct: " "
},
```

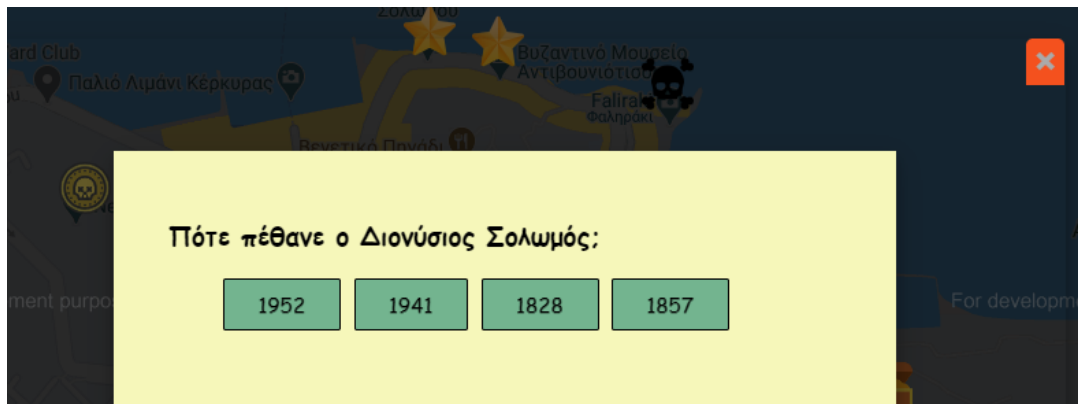
Μέσα σε αυτή την μέθοδο δίνουμε την εντολή η κάθε απάντηση που θα εμφανίζεται στον χρήστη να είναι ένα κουμπί. Το κάναμε κουμπί για να γίνεται πιο εύκολα ο έλεγχος της απάντησης εάν είναι σωστή ή όχι, με την βοήθεια της μεθόδου onClick. Επίσης, στην className, γίνεται έλεγχος της μεταβλητής isActive. Αν αυτή η μεταβλητή είναι αληθής τότε καλείται στο css αρχείο ο κώδικας για το κουμπί να γίνεται πράσινο στο χρώμα εάν η απάντηση είναι ορθή αλλιώς παραμένει κώδικα με τον άλλο κώδικα.

Στο κάθε κουμπί που θα επιστρέφεται (τόσα όσα και οι απαντήσεις του JSON αρχείου) έχουμε ένα κλειδί και την μέθοδο onClick. Τα κλειδιά βοηθούν στο React να προσδιορίσει ποια στοιχεία έχουν αλλάξει στον πίνακα και με την άλλη μέθοδο καλούμε την συνάρτηση "Score" με όρισμα τις απαντήσεις για τον έλεγχό τους. Με την εντολή {answers}, απλώς εμφανίζονται τα στοιχεία του πίνακα.

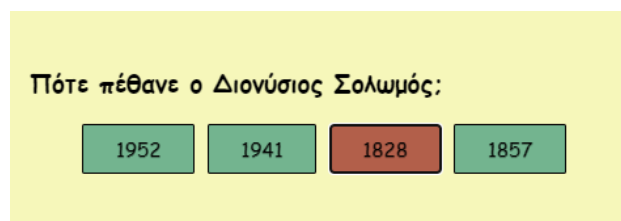
```
{Questions[0].answers.map((answers, index) => (
  <button
    key={index}
    className={this.state.isActive ? "ButtonAnswer1" : "ButtonAnswer"}

    onClick={() => this.Score(answers)} //pass the answers into the Score
function
  >
    {answers}
  </button>
))}
```

Και έτσι κλείνουν τα modals των ερωτήσεων και το αποτέλεσμα που παίρνουμε είναι το εξής,



Και τα χρώματα, κόκκινο για λάθος απάντηση



Και πράσινο για σωστή,

Πότε πέθανε ο Διονύσιος Σολωμός:

1952

1941

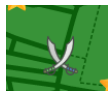
1828

1857

Έχουμε τώρα κάποιες περιπτώσεις μηνυμάτων που εμφανίζονται με διαφορετικό τρόπο από τι τα υπόλοιπα.

```
<div className="TestScore">...  
  /* Version 2 => Dec  
  <div className="TestScore1">...  
    /* Final Score  
    /* Version 1 => You  
  <div className="FinalDestination">...  
    /* Version 2 => You  
  <div className="FinalDestination1">...  
</Map>
```

Σε ένα συγκεκριμένο σημείο του χάρτη, στο «Πεντοφάναρο», με εικονίδιο τα πειρατικά σπαθιά,



έχουμε δύο επιλογές. Έχουμε δηλώσει μία συνάρτηση που ελέγχει τους βαθμούς του παιχνιδιού εάν είναι αρκετοί για να εμφανιστεί ένα μήνυμα αλλιώς μια ερώτηση και αυτή η συνάρτηση είναι η “testScoreBattle”.

Οπότε το modal της περίπτωσης που το σκορ είναι αρκετό είναι ίδιο με αυτά που δηλώσαμε πριν με τις ερωτήσεις. Το μόνο που αλλάζουν είναι οι μεταβλητές, βάζουμε αυτές που έχουν σχέση με την συνάρτηση και το Modal αυτό.

```
<div className="TestScore">  
  <Modal  
    className="hi"  
    show={this.state.testScore}  
    marker={this.state.activeMarker}  
    onHide={this.testScoreBattle}  
  >  
    <Modal.Header className="modal-header">  
      <button className="close" onClick={this.testScoreBattle}>  
        <span aria-hidden="true">&times;</span>  
      </button>  
    </Modal.Header>  
    <Modal.Body className="modal-content">  
      <h3>{Questions.map(q => q.question4)}</h3>  
      <ul>  
        {Questions[4].answers.map((answers, index) => (  
          <button
```

```

        key={index}
        className={this.state.isActive ? "ButtonAnswer1" : "ButtonAnswer"}
        onClick={() => this.Score(answers)}
      >
        {answers}
      </button>
    )})
  </ul>
</Modal.Body>
<Modal.Footer></Modal.Footer>
</Modal>
</div>;

```

Στην άλλη περίπτωση που δεν θέλουμε να εμφανίζονται οι ερωτήσεις αλλά ένα μήνυμα, έχουμε το ίδιο modal με τα αρχικά που συναντήσαμε, των απλών πληροφοριών. Απλώς είναι πάλι διαφορετικές οι μεταβλητές, οι συναρτήσεις και το μήνυμα που εμφανίζεται.

```

<div className="TestScore1">
  <Modal
    className="hi"
    show={this.state.testScore1}
    marker={this.state.activeMarker}
    onHide={this.testScoreBattle}
  >
    <Modal.Header className="modal-header">
      <button className="close" onClick={this.testScoreBattle}>
        <span aria-hidden="true">&times;</span>
      </button>
    </Modal.Header>
    <Modal.Body className="modal-content">
      <h3>Λυπάμαι.. </h3>
      <p>Μόλις σε αιχμαλώτισαν οι πειρατές! Ξαναδοκίμασε</p>
      <img className="skeleton" src={skeleton} alt="show angry skeleton"></img>
    </Modal.Body>
    <Modal.Footer></Modal.Footer>
  </Modal>
</div>;

```



Αντίστοιχα και στον τελικό προορισμό με εικονίδιο ένα σεντούκι, έχουμε δύο μηνύματα που καλούν την συνάρτηση “finaldestination”. Όπως αναφέραμε σε αυτή την συνάρτηση ελέγχουμε εάν το σκορ είναι πάνω από πενήντα (δηλαδή έχουν απαντηθεί όλες οι ερωτήσεις σωστά) τότε εμφανίζεται ένα μήνυμα ολοκλήρωσης του παιχνιδιού με κομφετί και ένα κουμπί που επιτρέπει στον παίκτη πατώντας το να επιστρέψει στο αρχικό μενού. Αλλιώς εμφανίζεται μήνυμα ότι πρέπει να συνεχιστεί το παιχνίδι για την ολοκλήρωσή του.

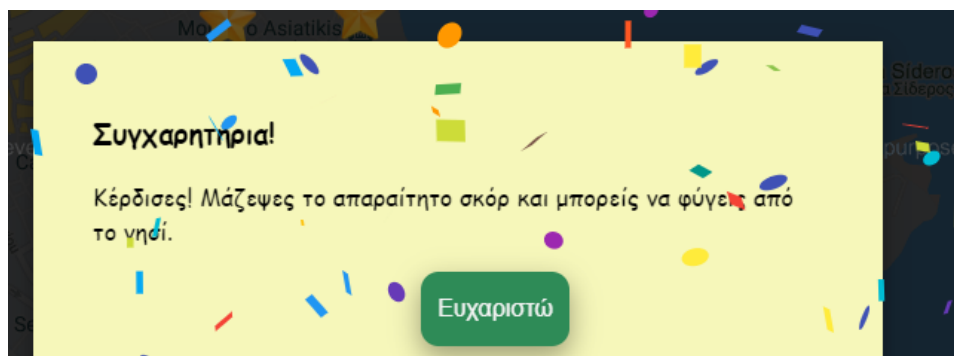
Έτσι στην πρώτη περίπτωση , το modal παραμένει ίδιο , αλλάζουν μόνο οι συναρτήσεις , οι μεταβλητές και προστίθεται τα κομφετί και το κουμπί.

```
<div className="FinalDestination">
  <Modal
    className="hi"
    show={this.state.finale}
    marker={this.state.activeMarker}
    onHide={this.finaldestination}
  >
    <Modal.Header className="modal-header">
      <button className="close" onClick={this.finaldestination}>
        <span aria-hidden="true">&times;</span>
      </button>
    </Modal.Header>
    <Modal.Body className="modal-content">
      <h3>Συγχαρητήρια!</h3>
      <p>
        Κέρδισες! Μάζεψες το απαραίτητο σκορ και μπορείς να φύγεις από το νησί.
      </p>
      { /*          Add some confetti          */ }
      <Confetti
        config={{
          angle: "360",
          spread: 360,
          startVelocity: "21",
          elementCount: "160",
          dragFriction: "0.05",
          duration: "4080",
          stagger: "9",
          width: "19px",
          height: "31px",
          perspective: "208px",
          colors: ["#a864fd", "#29cdff", "#78ff44", "#ff718d", "#fdff6a"]
        }}
      />
      <button className="HomePage" onClick={this.Go}>
        Ευχαριστώ
      </button>
    </Modal.Body>
    <Modal.Footer></Modal.Footer>
  </Modal>
</div>;
```

Για τα κομφετί , βοήθησε το παράδειγμα της βιβλιοθήκης npm “react confetti” και για το σχέδιο , να μην είναι τυχαίο αλλά στρόγγυλα κομφετί η σελίδα του danile-lundin “react dom confetti”. Όπου σε αυτούς

τους κώδικες εξηγούνται πως γίνονται οι αλλαγές στα κομμάτια του ηρω, στα χρώματα, την ταχύτητα που πέφτουν και το σχήμα τους.

Το τελικό αποτέλεσμα,



Στην άλλη περίπτωση είναι το ίδιο modal με τα υπόλοιπα που περιέχουν μόνο ένα απλό μήνυμα και για αυτό δεν χρειάζεται να δείξουμε τον κώδικα.

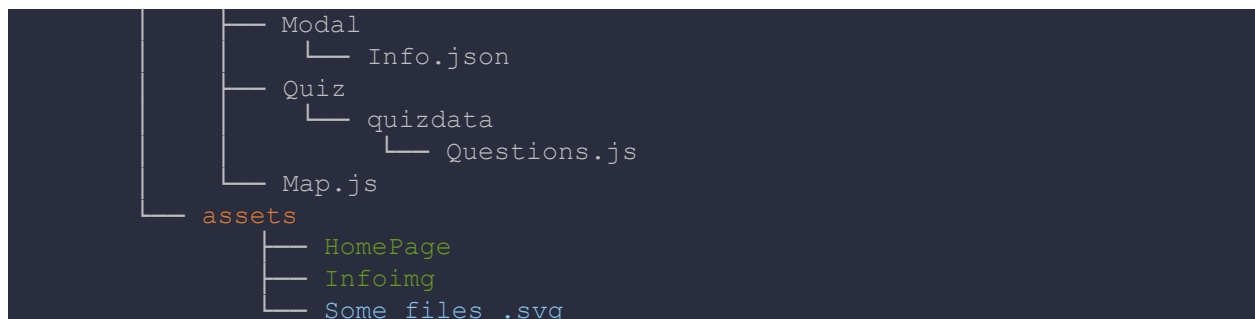
Έτσι μένουμε στην τελευταία εντολή αυτού του μεγάλου αρχείου την, Score.

Για να έχουμε μία βαθμολογία επάνω από τον χάρτη, την δηλώνουμε εκτός αυτού αλλά μέσα στην κλάση μας. Αυτή η απλή εντολή λέει στο πρόγραμμα να εμφανίσει την λέξη «score» και να εμφανίσει το περιεχόμενο της μεταβλητής “score” που έχουμε δηλώσει στον κατασκευαστή στην αρχή με την τιμή μηδέν. Αυτή η μεταβλητή καλείται σε διάφορες συναρτήσεις στο αρχείο και η τιμή της μεταβάλλεται συνεχώς. Κάθε αλλαγή που συμβαίνει, με την setState ενημερώνεται ο κατασκευαστής και έτσι εμφανίζονται οι αλλαγές αυτές σε πραγματικό χρόνο.

```
</Map>
<div className="Score_Text">
  <h1>Score : {this.state.score}</h1>
</div>
```

Αφού τελειώσαμε με την εξήγηση του κώδικα των αρχείων της εφαρμογής μας έμεινε μόνο να εξηγήσουμε τι είναι ο φάκελος “assets”.

```
Corfu-escape-adventure/
├── README.md
├── node_modules/
├── package.json
├── .gitattributes
├── .gitignore
├── public/
│   ├── index.html
│   └── favicon.ico
└── src/
    ├── App.css
    ├── App.js
    ├── Index.js
    ├── Index.css
    ├── mapStyles.js
    ├── serviceWorker.js
    └── Components
```



Ο φάκελος αυτός περιέχει αρχεία με κατάληξη .svg και δύο εμφωλευμένους φακέλους.

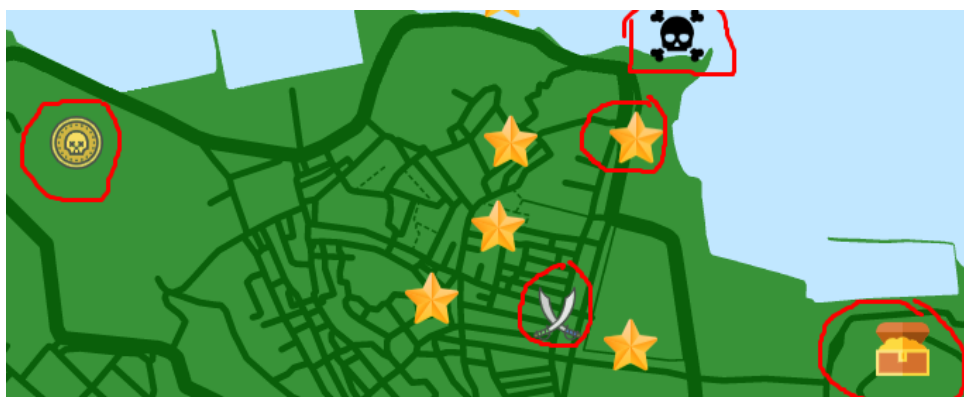
Τα αρχεία svg (Scalable Vector Graphics file) , είναι μια διανυσματική γραφική μορφή βασισμένη σε eXtensible Markup Language (XML) για τον Ιστό και άλλα περιβάλλοντα για να περιγράψουν τον τρόπο εμφάνισης της εικόνας. Αυτό το αρχείο μπορεί να κλιμακωθεί σε διαφορετικά μεγέθη χωρίς απώλεια ποιότητας - με άλλα λόγια, η μορφή είναι ανεξάρτητη από την ανάλυση και χρησιμοποιείται συνήθως από το DOM, Css και Javascript. Έτσι χρησιμοποιήθηκε αυτή η μορφή αρχείων σαν εικόνες για τα σημεία του χάρτη.

Το κάλεσμα των αρχείων αυτών έγινε στο “Map.js” , στην αρχή του αρχείου, στις αρχικοποιήσεις.

```
var markericon = require("../assets/star.svg");
```

Έτσι κάθε φορά που θα βλέπει ο κώδικας την μεταβλητή “markericon”, ή τις αντίστοιχες μεταβλητές με άλλη ονομασία θα καλούνται τα αρχεία SVG. Αυτά τα αρχεία είναι μεγάλα αρχεία και αναλόγως την διαμόρφωση του «εικονιδίου» υπάρχουν και οι αντίστοιχες εντολές.

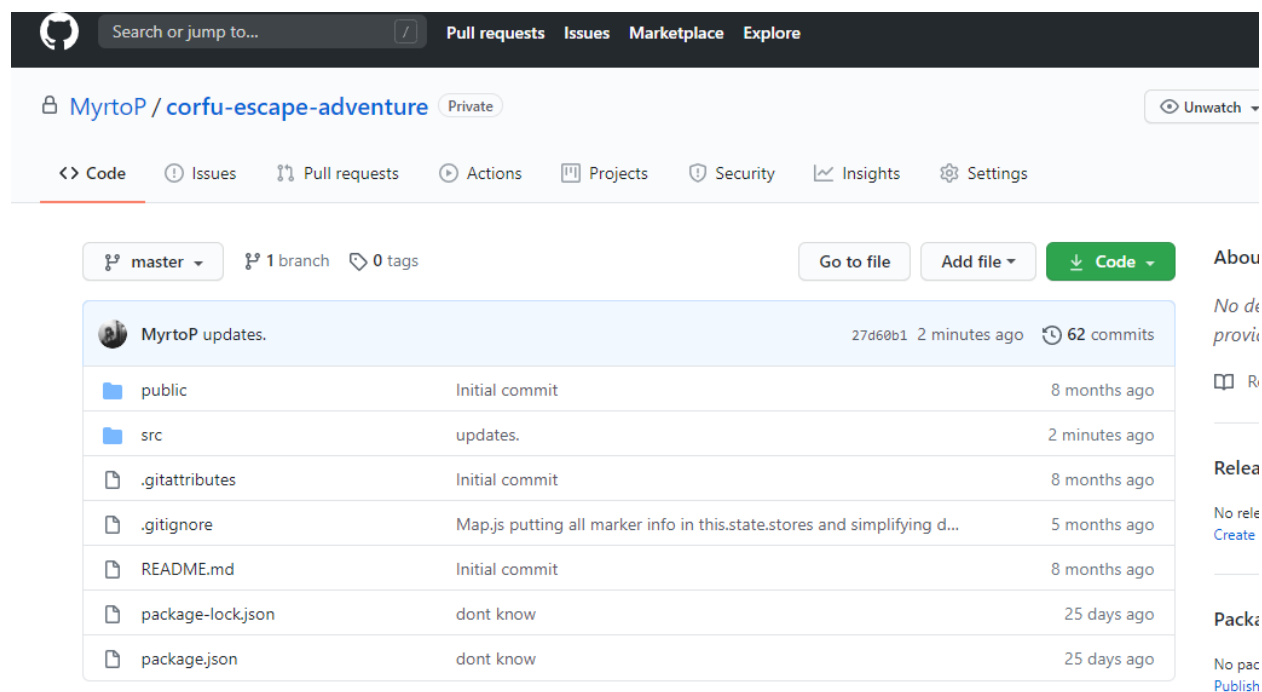
Μπορεί κάποιος να προσθέσει στο πρόγραμμα του μια εικόνα svg και αυτόματα το περιβάλλον κωδικοποίησης αναγνωρίζει τον τύπο του αρχείου αυτού και το εμφανίζει με την κανονική του μορφή, ένα XML κώδικα. Αλλά όταν καλούμε το αρχείο αυτό με την παραπάνω εντολή στην οθόνη επιστρέφεται σαν ένα απλό εικονίδιο. Η νεκροκεφαλή, το νόμισμα , τα αστέρια, τα πειρατικά σπαθιά και το μπαούλο είναι όλα τέτοια αρχεία.



Οι άλλοι δύο φάκελοι “HomePage” και “Infoimg” , είναι φάκελοι που περιέχουν εικόνες που χρησιμοποιούμε στο πρόγραμμα.



Τέλος, αναφέρουμε πως η εφαρμογή αυτή δημιουργήθηκε τοπικά στον υπολογιστή, με την χρήση του Node.js τερματικού και το πρόγραμμα VSCode για τον κώδικα. Οπότε για να μην χαθεί το περιεχόμενο αξιοποιήθηκε το “*GithubDesktop” ως αποθετήριο για ενδιάμεσες εκδόσεις . Δηλαδή δημιουργήθηκε ένας φάκελος του project όπου ενημερώνει το *gist του διαδικτυακού “*Github” σε κάθε αλλαγή που πραγματοποιείτε. Αυτός ο φάκελος υπάρχει στο διαδίκτυο, ώστε να μπορεί να το μοιραστεί και σε άλλους προγραμματιστές.



Search or jump to... / Pull requests Issues Marketplace Explore

MyrtoP / corfu-escape-adventure Private Unwatch

<> Code ! Issues 🔗 Pull requests ▶ Actions 📁 Projects 🔒 Security 📈 Insights ⚙ Settings

master 1 branch 0 tags Go to file Add file Code

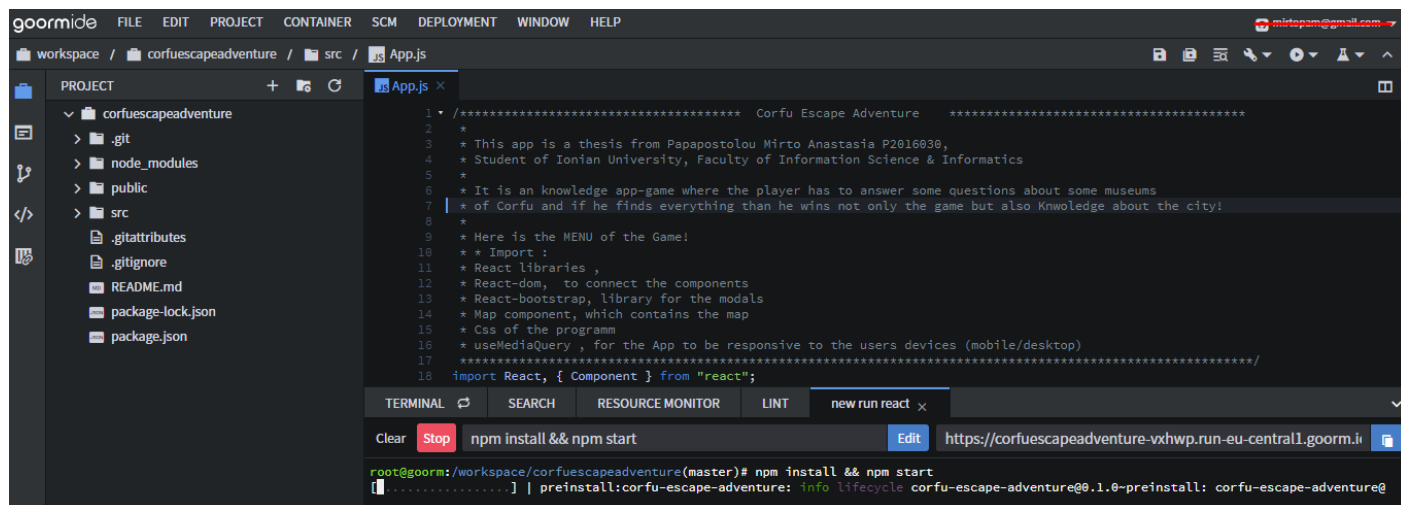
MyrtoP updates.	27d60b1 2 minutes ago	62 commits
public	Initial commit	8 months ago
src	updates.	2 minutes ago
.gitattributes	Initial commit	8 months ago
.gitignore	Map.js putting all marker info in this.state.stores and simplifying d...	5 months ago
README.md	Initial commit	8 months ago
package-lock.json	dont know	25 days ago
package.json	dont know	25 days ago

About No de provi

Relea No rele Create

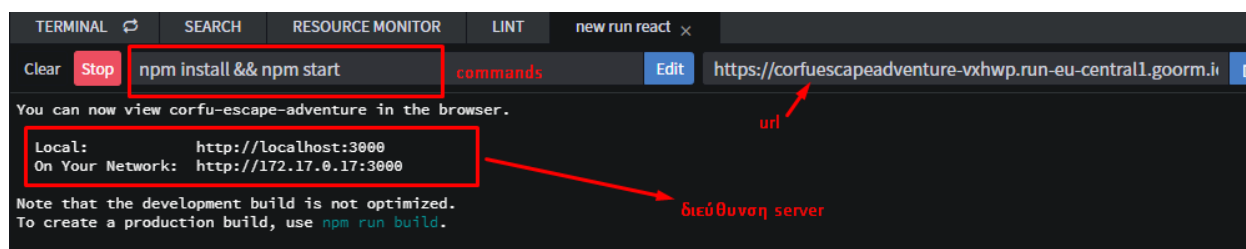
Pack: No pac Publish

Επίσης, χρησιμοποιήθηκε η πλατφόρμα “GoormIDE”. Σε αυτή την πλατφόρμα «ανέβηκε» ο κώδικας και από τον ιστότοπο παραχωρήθηκε ένας σέρβερ , όπου παρέχει μια διεύθυνση url στο project μας για κοινοποίηση. Καθώς τρέχει ο σέρβερ μπορεί οποιοσδήποτε κατέχει αυτή την διεύθυνση να παίζει το παιχνίδι μας. Για αυτό ακριβώς το λόγο έγινε και η επιλογή της πλατφόρμας αυτής, άλλες σελίδες δεν παρείχαν έναν σέρβερ να φιλοξενεί το πρόγραμμα μόνο ένα url που μπορούσε να το δει μόνο ο προγραμματιστής.



The screenshot shows the goormide IDE interface. On the left, the 'PROJECT' pane displays the file structure of 'corfuescapeadventure', including folders like '.git', 'node_modules', 'public', 'src' and files like '.gitattributes', '.gitignore', 'README.md', 'package-lock.json', and 'package.json'. The main editor shows the 'App.js' file with a multi-line comment describing the 'Corfu Escape Adventure' app, its purpose, and the libraries used (React, React-dom, React-bootstrap, Map). The terminal at the bottom shows the command 'npm install && npm start' being executed, with output indicating the installation of 'corfu-escape-adventure@0.1.0-preinstall'.

Για να τρέξει το πρόγραμμα απλώς, πατάμε το “App.js” αρχείο να “τρέξει”, και αυτόματα το τερματικό του ιστότοπου τρέχει τις εντολές, `npm install & npm start`, για να μπορέσει να τρέξει το React πρόγραμμα και παράλληλα δίνεται η ηλεκτρονική διεύθυνση.



This screenshot focuses on the terminal output of the previous command. The terminal shows the command 'npm install && npm start' and the message 'You can now view corfu-escape-adventure in the browser.' Below this, two URLs are listed: 'Local: http://localhost:3000' and 'On Your Network: http://172.17.0.17:3000'. A red box highlights these URLs, and a red arrow points from the text 'διεύθυνση server' to the 'On Your Network' URL. Another red arrow points from the text 'url' to the URL in the address bar of the terminal window.

* Git , είναι ένα σύστημα διαχείρισης εκδόσεων αρχείων. Στόχοι του Git είναι να υποστηρίξει τη διανεμημένη παραγωγικότητα και την ταχύτητα. Το Git βοηθά τον χρήστη να διατηρήσει την υπευθυνότητα και την πατρότητα για κάθε δράση του, να διατηρεί την ακεραιότητα του έργου και να βελτιώνει τη συνολική επικοινωνία μεταξύ των συνεργατών.



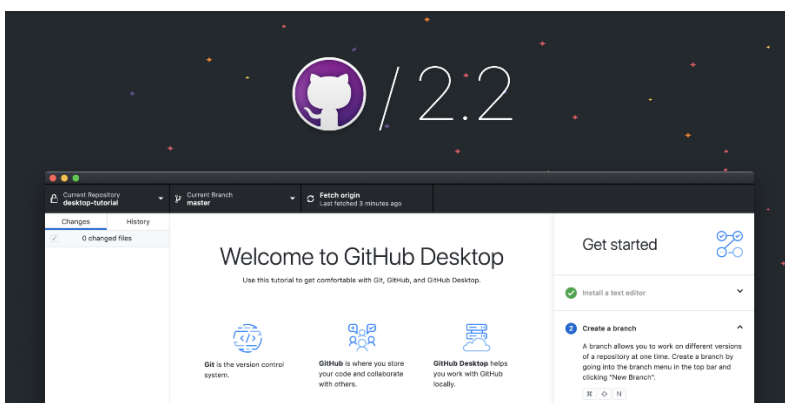
aimatus.com, “Using multiple Git accounts and SSH keys”

*Github, είναι ένας εξυπηρετητής που φιλοξενεί αποθετήρια και μια εύχρηστη διεπαφή ιστού. Υποστηρίζει όλες τις εντολές του git και παρέχει μερικές επιπλέον δυνατότητες. Επίσης παρέχει έλεγχο πρόσβασης και πολλές δυνατότητες συνεργασίας, όπως ανίχνευση σφαλμάτων, αιτήματα χαρακτηριστικών, διαχείριση εργασιών, και wikis για κάθε project.



somagnews.com, “GitHub Key Features Now Free For All Users”

*Desktop Github, απλοποιεί τη ροή εργασιών ανάπτυξης. Αντί να γράφουν οι προγραμματιστές εντολές του git, χρησιμοποιούν το Dekstop διότι προσφέρει μια «οπτική» εμπειρία, σαν να χρησιμοποιούν ένα απλό ιστότοπο από ένα τερματικό.



github.blog, “Getting started with Github Desktop”

*GoormIDE , είναι ένα διαδικτυακό εργαλείο cloud προγραμματισμού (διάθεση υπολογιστικών πόρων μέσω διαδικτύου). Παρέχει ένα περιβάλλον ανάπτυξης σε διάφορες γλώσσες όπως Java, Python, React και ούτω καθεξής, χωρίς καμία εγκατάσταση, μόνο σύνδεση στο διαδίκτυο. Προσφέρει επίσης ισχυρές δυνατότητες συνεργασίας, καθιστώντας εύκολο τον προγραμματισμό ταυτόχρονα με άλλους προγραμματιστές.



ide.goorm.io

Αξιολόγηση

Δημιουργήθηκαν δύο ερωτηματολόγια για την εργασία αυτή. Ένα στην αρχή πριν δημιουργηθεί το παιχνίδι για ιδέες (κεφάλαιο «Σχεδιαση») και ένα στο τέλος για εντυπώσεις και συμβουλές για πιθανές μελλοντικές βελτιώσεις.

Και τα δύο ερωτηματολόγια μοιράστηκαν σε πάνω από δέκα ανθρώπους και διεξήχθησαν ηλεκτρονικά μέσω της πλατφόρμας της Google, το “Google φόρμες”. Στην επικεφαλίδα του καθενός ερωτηματολογίου, υπάρχει μία μικρή ενημέρωση γραπτός για το θέμα του παιχνιδιού και για ποιο λόγο γίνονται αυτές οι ερωτήσεις. Η συμμετοχή των εθελοντών που ερωτώνται είναι ανώνυμη. Επίσης, οι παίκτες ήταν ηλικιών από 15 έως 50 ετών και τους κοινοποιήθηκε μία διεύθυνση url, από την πλατφόρμα “GoormIDE”, για να μπορούν να δουν την εφαρμογή διαδικτυακά χωρίς να χρειάζεται να εγκαταστήσουν ειδικά εργαλεία (π.χ. Node.js τερματικό).

Οι ερωτήσεις του δεύτερου ερωτηματολογίου,

1. Πρώτη εντύπωση (ως προς την εμφάνιση)

Κείμενο σύντομης απάντησης

2. Το μενού σου φάνηκε εύχρηστο ; (εάν υπήρχαν αρκετές εξηγήσεις για τον τρόπο λειτουργίας του παιχνιδιού)

☐ ΝΑΙ

☐ ΟΧΙ

☐ Άλλο...

3. Το στυλ του χάρτη ταιριάζει με έναν πειρατικό;

	1	2	3	4	5	
Καθόλου	☐	☐	☐	☐	☐	Αρκετά

4. Οι πληροφορίες είναι αρκετές ;

	1	2	3	4	5	
Καθόλου	☐	☐	☐	☐	☐	Αρκετά

5. Ήταν κατανοήσιμες και ευανάγνωστες;

	1	2	3	4	5	
Καθόλου	☐	☐	☐	☐	☐	Αρκετά

6. Οι ερωτήσεις ήταν αρκετές;

	1	2	3	4	5	
Καθόλου	☐	☐	☐	☐	☐	Αρκετά

7. Οι ερωτήσεις ήταν δύσκολες ; (εάν ήταν κατανοήσιμες και ευανάγνωστες)

	1	2	3	4	5	
Καθόλου	☐	☐	☐	☐	☐	Αρκετά

8. Προσκομεί κάποιος κάτι από το παιχνίδι ;

- ☐ Ναι
- ☐ Όχι
- ☐ Άλλο...

9. Πιστεύεις πως υπήρχε ελευθερία κίνησης μέσα στο παιχνίδι; (να επιλέγει ο παίκτης με ποια σειρά θα απαντάει τις ερωτήσεις)

	1	2	3	4	5	
Καθόλου	☐	☐	☐	☐	☐	Αρκετά

10. Σου φάνηκε εύκολο παιχνίδι;

	1	2	3	4	5	
Καθόλου	☐	☐	☐	☐	☐	Αρκετά

11. Θα ήθελες το παιχνίδι να παίζεται και σε κινητή συσκευή; (να κινείσαι και να απαντάς τις ερωτήσεις)

Κείμενο σύντομης απάντησης

12. Σου άρεσε το παιχνίδι; (εάν υπήρχε διάδραση, εμφάνιση, πλοκή κλπ.)

Κείμενο μακροσκελούς απάντησης

13. Υπήρχαν σφάλματα στο παιχνίδι;

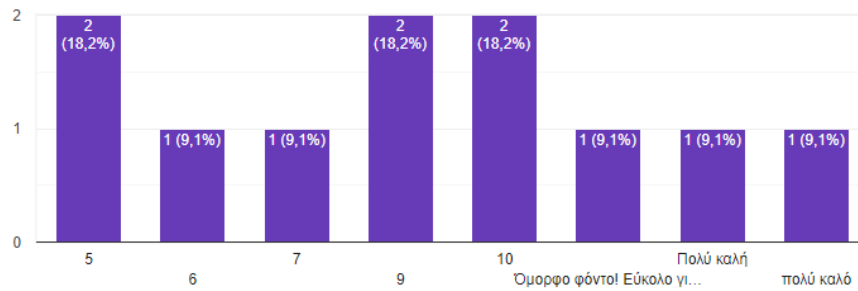
Κείμενο σύντομης απάντησης

14. Κάτι άλλο να προσθέσεις..;

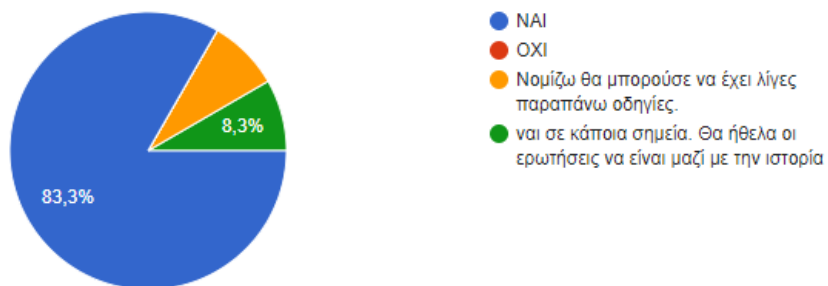
Κείμενο μακροσκελούς απάντησης

Οι απαντήσεις ,

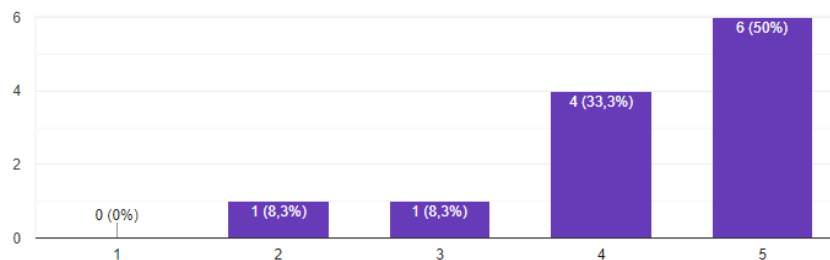
1. Πρώτη εντύπωση (ως προς την εμφάνιση)



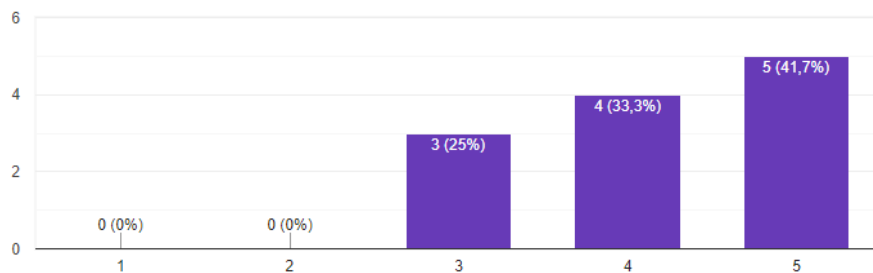
2. Το μενού σου φάνηκε εύχρηστο ; (εάν υπήρχαν αρκετές εξηγήσεις για τον τρόπο λειτουργίας του παιχνιδιού)



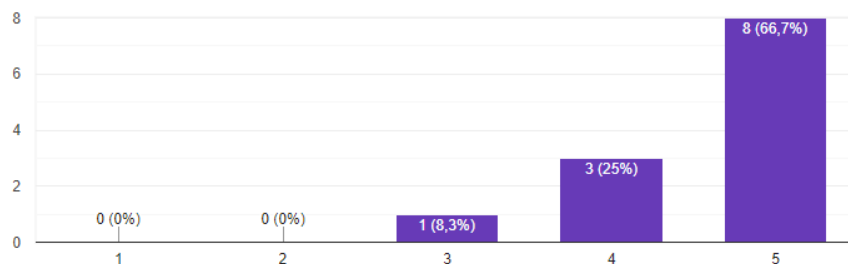
3. Το στυλ του χάρτη ταιριάζει με έναν πειρατικό;



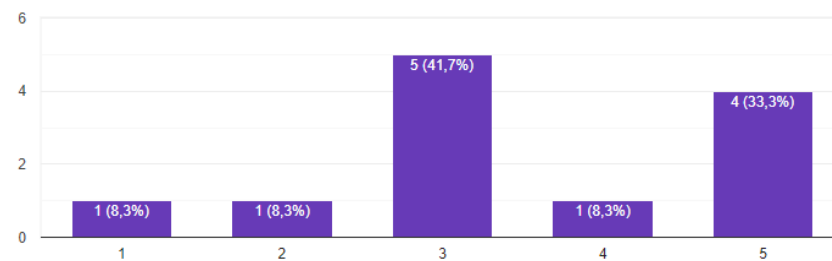
4. Οι πληροφορίες είναι αρκετές ;



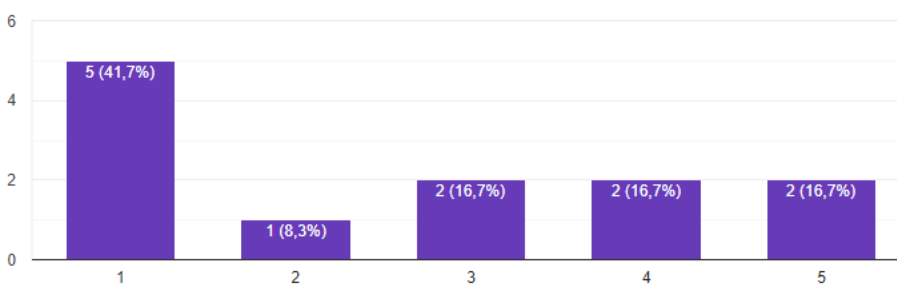
5. Ήταν κατανοήσιμες και ευανάγνωστες;



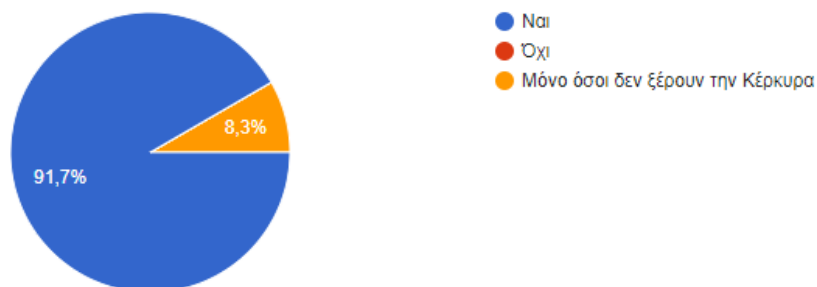
6. Οι ερωτήσεις ήταν αρκετές;



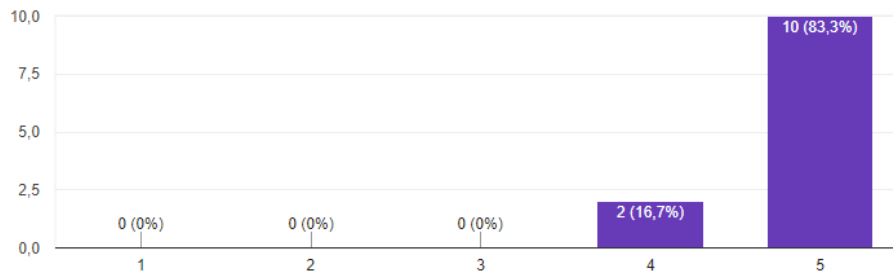
7. Οι ερωτήσεις ήταν δύσκολες ; (εάν ήταν κατανοήσιμες και ευανάγνωστες)



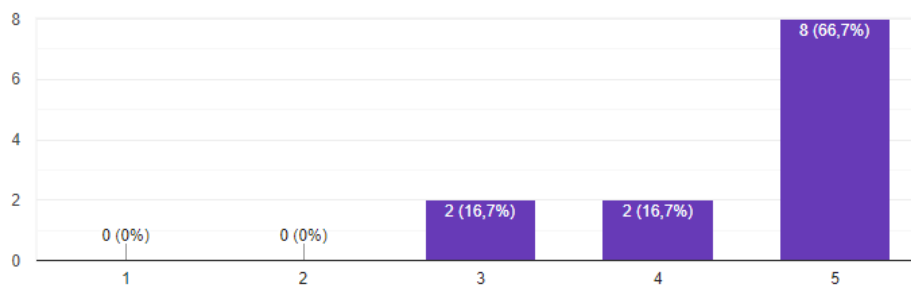
8. Προσκομεί κάποιος κάτι από το παιχνίδι ;



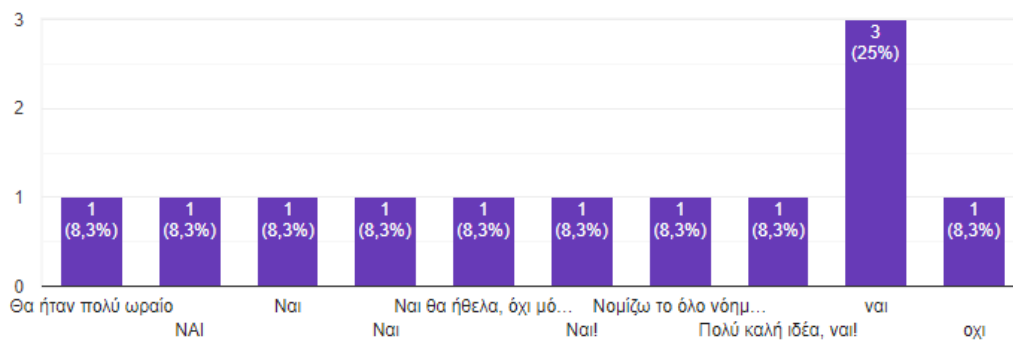
9. Πιστεύεις πως υπήρχε ελευθερία κίνησης μέσα στο παιχνίδι; (να επιλέγει ο παίκτης με ποια σειρά θα απαντάει τις ερωτήσεις)



10. Σου φάνηκε εύκολο παιχνίδι;



11. Θα ήθελες το παιχνίδι να παίζεται και σε κινητή συσκευή; (να κινείσαι και να απαντάς τις ερωτήσεις)



12. Σου άρεσε το παιχνίδι; (εάν υπήρχε διάδραση, εμφάνιση, πλοκή κλπ.)

- ΝΑΙ
- Ευχάριστο!
- Θα μπορούσε να υπάρχει περισσότερη πλοκή
- Συμπαθητικό
- ναι
- Ναι πολύ καλή προσπάθεια
- Ήταν πολύ ευχάριστο, έμαθα καινούρια πράγματα για το νησί και μου άρεσε η πλοκή
- ναι πολύ
- Πάρα πολύ! Έξυπνο και ενδιαφέρον ως προς τις γνώσεις που προσκομίζονται

- Διαδραστικό παιχνίδι με εξαιρετική εμφάνιση που σου προσελκύει το ενδιαφέρον να παίξεις και στην συνέχεια να αντλήσεις γνώσεις από αυτό.
- ναι

13. Υπήρχαν σφάλματα στο παιχνίδι;

- ΝΑΙ
- Δεν βρήκα κάποιο.
- ναι
- Κάποιες φορές δεν πρασίνιζε η σωστή ερώτηση
- ναι, κάποιες φορές ενώ απαντούσες σωστά και μέτραγε συν 10, η απάντηση ήταν κόκκινη και όχι πράσινη
- Το μενού ήταν κάτω αριστερά
- Δεν παρατήρησα κάποιο σφάλμα
- Ναι.
- δεν πρόσεξα κάτι
- Όχι!
- Όχι

14. Κάτι άλλο να προσθέσεις..;

- περισσότερες ερωτήσεις
- Θα μπορούσαν να μπουν περισσότερες ερωτήσεις για το Λιμάνι, την Παλιά Πόλη, τα καντούνια, την Παλαιά πόλη κ.α.
- θα ήθελα να είχε περισσότερες ερωτήσεις, ήταν πολύ ενδιαφέρον
- Εξαιρετική δουλειά!
- Ενδιαφέρον παιχνίδι ερωτήσεων - γνώσεων προς όλες τις ηλικίες

Όσο αφορά το δεύτερο ερωτηματολόγιο που είχε θέμα την αξιολόγηση του προσχεδίου οι χρήστες δήλωσαν ευχαριστημένοι για την εμφάνιση του αρχικού μενού και του χάρτη. Επιπλέον, βρήκαν τις ερωτήσεις ικανοποιητικές, εύκολες αλλά ταυτόχρονα μερικοί επιθυμούσαν να προστεθούν και άλλες. Αντίστοιχα, για τις πληροφορίες δήλωσαν πώς ήταν αρκετές, ευανάγνωστες, και αξιόλογες.

Το παιχνίδι φάνηκε ευχάριστο, εύκολο και δήλωσαν επιθυμία να το δουν σε φορητές συσκευές. Επίσης ήταν ευχαριστημένοι που υπήρχε ελευθερία επιλογής διαδρομής, μιας και το είχα ζητήσει στο πρώτο ερωτηματολόγιο. Τέλος, όσον αφορά για σφάλματα σε κάποιους εμφανιζόταν διαφορετικά διότι είχαν συσκευές διαφορετικής ανάλυσης και κάποιοι δήλωσαν πώς υπήρχε πρόβλημα με τα χρώματα των απαντήσεων των κουίζ.

ΠΡΟΒΛΗΜΑΤΑ & ΛΥΣΕΙΣ

Κατά την διάρκεια «χτισίματος» του παιχνιδιού “Corfu Escape Adventure” υπήρξαν κάποιες δυσκολίες. Οι δυσκολίες αυτές ήταν από κατανόησης μέχρι τεχνικών σφαλμάτων.

Το πρώτο σφάλμα που εμφανίστηκε ήταν το μήνυμα «Module not found: Can't resolve 'google-maps-react'» στο αρχείο “Map.js”. Επειδή το μήνυμα δηλώνει πως υπάρχει πρόβλημα με την βιβλιοθήκη αυτή, έγινε εγκατάσταση στο τερματικό του Node.js το πακέτο «google-maps-react» αλλά και στο προγραμματιστικό περιβάλλον VSCode με την εντολή, « npm install --save google-maps-react ».

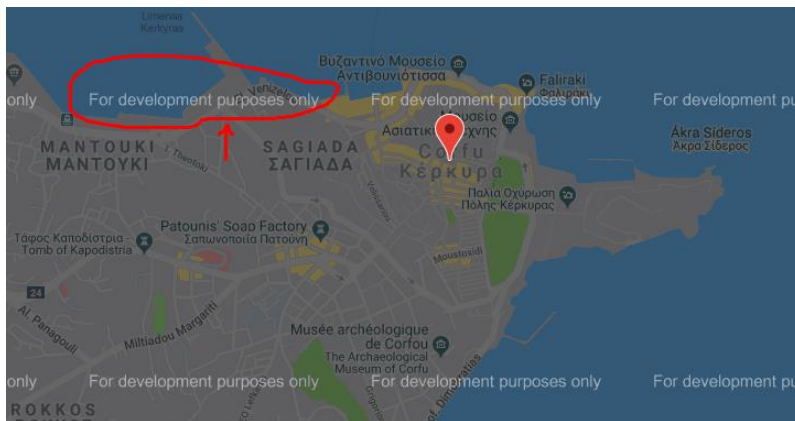
Μέχρι να γίνει καλή εμπέδωση του τί είναι η React και πώς ακριβώς λειτουργεί πήρε αρκετό χρονικό διάστημα το στήσιμο του βασικού κομματιού του παιχνιδιού, τον χάρτη.

Είχαμε στην διάθεσή μας το Api κλειδί αλλά όχι την κατανόηση στο πώς θα εμφανιστεί ο χάρτης. Συμβουλευτήκαμε τις πληροφορίες που παρείχε αυτό το ιστότοπος, scotch.io το άρθρο “ Rachael Njeri - React Apps with the Google Maps API and google-maps-react” για το πώς δουλεύει το “GoogleApiWrapper”.

Έπειτα αφού τοποθετήθηκε ο χάρτης και εμφανιζόταν στο localhost, εμφανιζόταν σε μία προκαθορισμένη πόλη της Αμερικής. Αυτό συνέβαινε από τη JavaScript google api βιβλιοθήκη και απλώς χρειαζόταν η αλλαγή των συντεταγμένων για την Κέρκυρα, με την εντολή “initialCenter”

```
<Map
  google={this.props.google}
  zoom={14}
  initialCenter={{ lat: 39.6249838, lng: 19.922346100000027 }}
>
```

Ένα από τα βασικά προβλήματα που αντιμετωπίστηκαν στην εργασία ήταν ότι ο χάρτης που παρέχει η βιβλιοθήκη της Google δεν εμφανίζεται όπως εμφανίζεται στο διαδίκτυο παραδείγματος χάριν , στο “Google Maps”. Εμφανιζόταν συνεχώς το μήνυμα που φαίνεται στη φωτογραφία,



Δηλαδή, για να εμφανίζεται ο χάρτης όπως επιθυμεί ο κάθε προγραμματιστής πρέπει να πληρώσει στην Google ένα ποσό. Επειδή στο παιχνίδι ο χάρτης θέλουμε να είναι σαν πειρατικός και έχουμε βάλει τον κώδικα να το κάνει αυτό, πήγαμε στον ιστότοπο, developer.google.com , στο εγχειρίδιο χρήσης στην ενότητα «Get Started with Google Maps Platform». Σε αυτή την σελίδα εξηγεί το πώς μπορεί ένας προγραμματιστής να προσθέσει έναν λογαριασμό από τον οποίο θα πληρώνει ένα ποσό για να μπορεί να

χρησιμοποιεί το API ενός χάρτη. Για κάποιο χρονικό διάστημα είναι δωρεάν, έτσι και ενεργοποιήθηκε ο λογαριασμός και ο χάρτης εμφανίζεται μόνο με το στυλ που έχει δηλωθεί στον κώδικα.

Για να μην ξεκινάει το παιχνίδι κατευθείαν από τον χάρτη, τοποθετήθηκε το αρχείο αυτό σε έναν φάκελο “Component” και στο αρχικό πρόγραμμα (“App.js”) δημιουργήθηκε ένα μενού. Το πρόβλημα σε αυτό το μενού ήταν ότι δημιουργήθηκαν τα κουμπιά αλλά όταν ο χρήστης πατούσε πάνω τους αυτά δεν λειτουργούσαν. Διαπιστώθηκε ότι το logo της εφαρμογής ήταν πολύ μεγάλο και κάλυπτε τα κουμπιά, μόλις το μέγεθος της εικόνας άλλαξε λειτουργούσαν ομαλά.

Το δεύτερο μεγαλύτερο πρόβλημα ήταν πώς θα γίνει με το πάτημα των κουμπιών να εμφανίζονται μηνύματα με μαύρο φόντο. Στην αρχή βρέθηκε σαν λύση της ιδέας αυτής, τα infowindow αλλά δεν ήταν το επιθυμητό αποτέλεσμα. Μετά από έρευνα ανακαλύφθηκαν τα modal popup. Πήρε αρκετό διάστημα για να γίνει αντιληπτός ο τρόπος λειτουργίας τους και η εφαρμογή τους, σύνδεση τους με τις onclick συναρτήσεις των κουμπιών. Μέσα από τα εξής άρθρα και εγχειρίδια χρήσης έγινε αντιληπτή η όλη διαδικασία και ο τρόπος λειτουργίας τους,

- [blog.bitsrc.io](https://blog.bitsrc.io/build-a-simple-modal-component-with-react/) , «Build a simple Modal Component with React» , Eden Ella
- [stackoverflow.com](https://stackoverflow.com/questions/54638462/react-opening-a-modal-on-google-maps-marker-click) , «(React) Opening a Modal on Google Maps Marker Click»
- [react-bootstrap.github.io](https://react-bootstrap.github.io/components/modal/) , «Modals»
- [material-ui.com](https://material-ui.com/components/modal/), «Modal»

Έπειτα έγινε η σύνδεση των κουμπιών με modal το πρώτο κουμπί με το play , θέλαμε να εμφανίζει τον χάρτη σε άλλη σελίδα και όχι σε μικρή έκδοση , σαν πληροφορία. Αυτό θα επιτυγχανόταν με μία δήλωση if. Έτσι σκεφτήκαμε να ελέγχουμε την εμφάνιση του χάρτη με μία μεταβλητή όπως τα modals, και εάν αυτή η τιμή της μεταβλητής είναι αληθής τότε θα καλούμε το αντικείμενο “Map.js”. Το if, δεν δούλεψε μέσα στην return της render μεθόδου και έτσι δοκιμάσαμε να την βγάλουμε εκτός, αρχικά σαν συνάρτηση. Αυτή η ιδέα δεν λειτούργησε και έτσι κρατήσαμε τον έλεγχο αυτό μέσα στην μέθοδο αλλά πριν την έξοδο εμφάνισης και δούλεψε.

```
render() {  
  //If "PLAY" button is pressed Go to the Game!  
  if (this.state.showComponent) {  
    return (  
      <div className="HOL0">  
        <Map />  
      </div>  
    );  
  }  
}
```

Υπήρξε δυσκολία στο πώς θα συνδεθούν τα σημεία του χάρτη το καθένα με δικό του αντίστοιχο modal. Αυτή η σκέψη λύθηκε με την χρήση της onclick μεθόδου μέσα στον πίνακα του κατασκευαστή για τα σημεία και την χρήση της στο κάλεσμα του κάθε μηνύματος. Κατόπιν, έπρεπε να γίνει η σωστή εισαγωγή του json αρχείου για να εμφανίζονται οι πληροφορίες μέσα στο κυρίως σώμα του modal, και οι επιθυμητές εικόνες. Αυτό τελικά ήταν εφικτό μέσω της μεθόδου map και το κάθε αντικείμενο που καλείται να δείχνει στην συγκεκριμένη επιθυμητή πληροφορία (x => x.information).

```
<Modal.Header className="modal-header">  
  {" "}  
  <button className="close" onClick={this.showasiastory}>
```

```

        <span aria-hidden="true">&times;</span>
      </button>
    </Modal.Header>
    <Modal.Body className="modal-content-info">
      <img
        className="ImageInfo"
        src={mouseio}
        alt="show museum profile"
      ></img>
      <h2>{museumInfo.map(x => x.title)}</h2>
      <p>{museumInfo.map(x => x.information)}</p>
    </Modal.Body>
    <Modal.Footer></Modal.Footer>
  </Modal>

```

Για να είναι πιο «όμορφη» η εμφάνιση του κειμένου και των φωτογραφιών, να υπάρχει το σωστό μέγεθος, η δυνατότητα να «σκρολάρει» ο χρήστης την πληροφορία για να την δει ολόκληρη γινόταν μόνο μέσω του css αρχείου.

Το τρίτο μεγαλύτερο πρόβλημα ήταν, το πώς θα γίνει η μέθοδος “onclick”, του κάθε modal να επιστρέφει τις απαντήσεις, σε μία σειρά με μορφή κουμπιών και κατόπιν να τις ελέγχει εάν είναι σωστές ή όχι μόλις πατηθούν από τον παίκτη.

Συνδυάστηκαν οι πληροφορίες των εξής ιστότοπων και εγχειριδίων χρήσης για την κατανόηση της λύσης του προβλήματος αυτού,

- upmostly.com, «Pass a Parameter Through onClick in React», October 15, 2019
- ozmoro.com, «How to pass a value to onClick event handler in React.js», Mon Jul 02, 2018
- stackoverflow.com, «how to compare state variables of 2 components in react?»
- reactjs.org, «Handling Events»

Δηλαδή, έπρεπε μέσα σε μία μέθοδο map, να καλείται ο κατάλληλος πίνακας (“Questions”) από το αρχείο JSON με την αντίστοιχη θέση της αντίστοιχης ερώτησης (“Questions[4]”) και ο συγκεκριμένος εμφωλευμένος του πίνακας (“answers”) (βήμα 1). Έπειτα μέσα στην μέθοδο αυτή για να επιστρέφεται το περιεχόμενο των πινάκων σαν κουμπιά έπρεπε να μπουν μέσα σε tags, που δηλώνουν ότι είναι κουμπιά και να εμφανίζεται (“{answers}”) (βήμα 2).. Για να είναι τα κουμπιά σε μία σειρά χρειαζόταν από την HTML, η κατηγορία (βήμα 3) και τέλος για να γίνεται ο έλεγχος των απαντήσεων αυτών των κουμπιών μία “onclick” μέθοδος που θα καλεί μία συνάρτηση για την «δουλειά» αυτή (βήμα 4). Αλλά για να δουλέψει αυτή η συνάρτηση τελικά έπρεπε να περασθούν σαν ορίσματα τα αντικείμενα των πινάκων (βήμα 5).

```

<h3>{Questions.map(q => q.question4)}</h3>

3. <ul>
  1. {Questions[4].answers.map((answers, index) => (
  2. <button
      key={index}
      className="ButtonAnswer"
  4, 5.   onClick={() => this.Score(answers)}

```

```

      >
      2.      {answers}
    </button>    )}}    </ul>

```

Έπειτα έπρεπε απλώς στις συναρτήσεις “Score” και “ testScoreBattle ”, να γίνονται οι σωστοί έλεγχοι των απαντήσεων και αντίστοιχα της βαθμολογίας για να εμφανίζεται είτε ένα μήνυμα είτε μια ερώτηση.

Για την πρώτη συνάρτηση , να δέχεται τα αντικείμενα του πίνακα (“index”) και να ενημερώνεται το σκορ. Αυτό θα επιτυγχανόταν με την “setState”, εξού και το αποτέλεσμα,

```

Score = index => {
  if (index === "1857") {
    this.setState({
      score: this.state.score + 10,

```

Για την δεύτερη συνάρτηση, η μόνη λύση για την εμφάνιση διαφορετικών μηνυμάτων ήταν με την συνθήκη «αν» και να γίνεται να καλείται η πρώτη συνάρτηση για τον έλεγχο των απαντήσεων. Άρα σαν πέρασμα βάλαμε πάλι τα αντικείμενα του πίνακα “answers” ώστε στο κάλεσμα της συνάρτησης “Score”, να περνάει το συγκεκριμένο πάτημα κουμπιού , δηλαδή η απάντηση του παίκτη για τον έλεγχο.

```

testScoreBattle = index => {
  //If not enough score
  if (this.state.score <= 30 && this.state.score >= 0) {
    this.setState(prevState => ({
      testScore1: !prevState.testScore1
    }));
  } else {
    //show an extra quiz question
    this.setState(prevState => ({
      testScore: !prevState.testScore
    }));
    //check the answer
    this.Score(index);
  }
};

```

Το τελευταίο πρόβλημα της εργασίας είναι η αλλαγή των χρωμάτων των κουμπιών των απαντήσεων. Δηλαδή άμα η απάντηση του παίκτη είναι ορθή να γίνεται πράσινο το κουμπί αλλιώς κόκκινο για να γνωρίζει εάν αυτό που διάλεξε ως απάντηση είναι σωστή ή όχι. Αυτό το πρόβλημα λύθηκε με την χρήση μίας μεταβλητής “isActive”. Την μεταβλητή αυτή την δηλώνουμε στον κατασκευαστή και έπειτα μέσα στο κάθε κουμπί θα κάνουμε έλεγχο εάν είναι αληθής ή όχι για να ανοίγει η αντίστοιχη κλάση στο css αρχείο.

```

<button
  key={index}
  className={this.state.isActive ? "ButtonAnswer1" : "ButtonAnswer"}
  onClick={() => this.Score(answers)}

```

Στην συνάρτηση, αντί για να αλλάζουμε απευθείας την τιμή σε αληθές-ψευδές, το κάνουμε με την “prevState”. Διότι έβγαζε το εξής πρόβλημα, άμα πατούσε ο παίκτης την σωστή απάντηση, άλλαζε το κουμπί στο σωστό χρώμα αλλά παρέμενε η τιμή της isActive, και όλα τα κουμπιά άμα πατιόντουσαν το χρώμα έμενε πράσινο.

```
Score = (index, isActive) => {  
  if (index === "1857") {  
    this.setState(prevState => ({  
      score: this.state.score + 10,  
      isActive: !prevState.isActive  
    }));  
  }  
}
```

Βέβαια ακόμα δεν είναι το επιθυμητό αποτέλεσμα , αλλά είναι πιο κοντά στην λύση.

Αρχική ιδέα

Η ιδέα ξεκίνησε με την επιθυμία να δημιουργηθεί ένα ηλεκτρονικό εκπαιδευτικό παιχνίδι σε φορητές συσκευές. Το παιχνίδι αυτό θα περιείχε ερωτήσεις και πληροφορίες για ένα θέμα. Το θέμα που επιλέχθηκε ήταν η πόλη της Κέρκυρας και συγκεκριμένα τα ιστορικά της μνημεία. Έτσι, ήρθε η ιδέα για να είναι το θέμα αυτό πιο διασκεδαστικό στους παίκτες η υπό αναπτυσσόμενη εφαρμογή να περιέχει έναν χάρτη με τα σημεία των μνημείων. Κάποια σημεία θέλαμε να περιέχουν πληροφορίες σε σχέση με την επιλεγόμενη τοποθεσία του παίκτη, άλλα να περιέχουν ερωτήσεις και ίσως και μίνι παιχνίδια, παραδείγματος χάριν τρίλιζα.

Τελικό Αποτέλεσμα

Το συμπέρασμα είναι ότι έγινε μια προσπάθεια να δημιουργηθεί αυτό το παιχνίδι, όμως έγινε μόνο η αρχή και ευελπιστούμε να προχωρήσει η εξέλιξη και η ανάπτυξη του στο προσεχές μέλλον σε ένα ολοκληρωμένο παιχνίδι εκμάθησης.

Χρησιμοποιήθηκε μια νέα γλώσσα προγραμματισμού, η «React», η οποία δεν προτιμάται συνήθως από τους προγραμματιστές για την δημιουργία παιχνιδιών αυτού του είδους διότι χρησιμοποιείται κυρίως για την κατασκευή ιστοσελίδων. Όμως, μέσα από την εφαρμογή της στο συγκεκριμένο παιχνίδι και αντιμετωπίζοντας τις δυσκολίες που συναντήσαμε μπορέσαμε να την μάθουμε και να μπορέσουμε να την χρησιμοποιήσουμε όσο γίνεται πιο ωφέλημα για τον στόχο μας.

Αυτή η προσπάθεια συντελεί στην ανάπτυξη και διάδοση των παιχνιδιών με βάση την χρήση τοποθεσίας και σοβαρού σκοπού. Καλό είναι να δημιουργούνται περισσότερα παιχνίδια για μάθηση σε συνδυασμό με διασκεδαστικά εργαλεία, πέραν των κλασικών βιντεοπαιχνιδιών για την απλή ψυχαγωγία.

Ξεκινώντας την δημιουργία του παιχνιδιού καθώς και σε όλη του την πορεία συναντώντας τα προβλήματα καταλάβαμε ότι για να τα ξεπεράσουμε έπρεπε ταυτόχρονα να ψάχνουμε στο διαδίκτυο για πληροφορίες τις οποίες προσαρμόζαμε στη συγκεκριμένη εργασία. Έπρεπε να μελετήσουμε την γλώσσα, τα εργαλεία της, για να μπορέσουμε να την εφαρμόσουμε. Ο χρόνος σε σχέση με τα προβλήματα που αντιμετωπίσαμε δεν ήταν ο απαιτούμενος και για αυτό δεν ολοκληρώθηκε το παιχνίδι μας.

Προσωπικά, αποκόμισα από το θεωρητικό κομμάτι τι είναι ηλεκτρονική μάθηση και πως μπορεί να επιτευχθεί μέσω ενός ηλεκτρονικού παιχνιδιού. Παράλληλα έμαθα και στην πράξη πως να δημιουργώ ηλεκτρονικά παιχνίδια που ανήκουν στις κατηγορίες, location based games και serious games. Σημαντικό θα ήταν να αναφερθεί πως μάθαμε να ερευνούμε, οργανώνουμε και να σχεδιάζουμε το θεωρητικό υπόβαθρο μας ώστε να μπορούμε να ψάξουμε τις κατάλληλες τεχνολογίες ώστε να μάθουμε να τις χειριζόμαστε για την δημιουργία της αρχικής μας ιδέας.

Πρακτικές δυσκολίες που συναντήσαμε ήταν κυρίως στην υλοποίηση του κώδικα. Η γλώσσα προγραμματισμού που χρησιμοποιήθηκε ήταν άγνωστη και έπρεπε να μαθαίνουμε παράλληλα με το «χτίσιμο» του προγράμματος τη γλώσσα και τα εργαλεία της.

Τελικά, σε σχέση με την αρχική ιδέα αυτό που μπόρεσε και υλοποιήθηκε είναι το ότι εμφανίζεται ένας χάρτης, με τα σημεία, τις πληροφορίες, τις ερωτήσεις και την βαθμολογία. Δυστυχώς, η εφαρμογή δεν ολοκληρώθηκε όπως θα επιθυμούσαμε λόγω τεχνικών προβλημάτων. Θα επιθυμούσαμε να λειτουργούσε σε κινητά τηλέφωνα, να μπορεί ο χρήστης να κάνει σύνδεση, να μιλάει με άλλους παίκτες, να

περιέχονται μίνι παιχνίδια και να υπάρχουν ανταμοιβές στην πραγματική ζωή (το παράδειγμα με τον καφέ).

Τι αποκόμισαν οι Χρήστες

Οι χρήστες ενθουσιάστηκαν με την ιδέα ενός παιχνιδιού ανεξαρτήτως εάν ήταν εκπαιδευτικό ή όχι. Εκτίμησαν την προσπάθεια και θέλανε να βοηθήσουν στην εξέλιξη του και την βελτίωσή του παρέχοντας ιδέες και λύσεις. Κάποιοι από τους χρήστες δεν γνωρίζουν από πληροφορική και δεν τους ενδιέφερε το τεχνικό κομμάτι και οι λεπτομέρειες, απλώς να διασκεδάσουν και να μάθουν μέσω του προσχεδίου αυτού. Για αυτό και ζητήσαν περισσότερες ερωτήσεις στην εφαρμογή για να μπορέσουν να γνωρίσουν την Κέρκυρα ακόμα και αν δεν την έχουν επισκεφθεί.

Οι χρήστες οι οποίοι γνωρίζουν πληροφορική, αναφέρθηκαν πιο πολύ στο τεχνικό κομμάτι και εστίασαν την προσοχή τους στα λειτουργικά τμήματα της εφαρμογής. Οι ιδέες τους ήταν ανάμεικτες και από ψυχαγωγικά παιχνίδια και από εκπαιδευτικά, με αποτέλεσμα να είναι πιο αυστηροί κριτές.

ΒΙΒΛΙΟΓΡΑΦΙΑ

ΘΕΩΡΙΑ

- Kostas, Vassilakis & Orestis, Charalampakos & Glykokokalos, Georgios & Kontokalou, Persefoni & Kalogiannakis, Michail & Vidakis, Nikolas. (2017). Learning History through Location-Based Games: The Fortification Gates of the Venetian walls of the city of Heraklion.
- Jacob, João & da Silva, Hugo & Coelho, António & Rodrigues, Rui. (2012). Towards Location-based Augmented Reality games. *Procedia Computer Science*. 15. 318–319. 10.1016/j.procs.2012.10.093.
- Jacob, João & Coelho, António. (2011). Issues in the Development of Location-Based Games. *Int. J. Computer Games Technology*. 2011. 10.1155/2011/495437.
- Andone, Ionut & Błaszkiwicz, Konrad & Böhmer, Matthias & Markowetz, Alexander. (2017). Impact of Location-based Games on Phone Usage and Movement: A Case Study on Pokémon GO. 10.1145/3098279.3122145.
- Matyas, Sebastian & Matyas, Christian & Schlieder, Christoph & Kiefer, Peter & Mitarai, Hiroko & Kamata, Maiko. (2008). Designing location-based mobile games with a purpose: Collecting geospatial data with cityExplorer. *Proceedings of the 2008 International Conference on Advances in Computer Entertainment Technology, ACE 2008*. 244-247. 10.1145/1501750.1501806.
- Ejsing-Duun, Stine. (2011). Location-based Games: From Screen to Streets.
- Susi, Tarja & Johannesson, Mikael & Backlund, Per. (2015). Serious Games - An Overview.
- Mouaheb, Houda & Fahli, Ahmed & Moussetad, Mohamed & Eljamali, Said. (2012). The Serious Game: What Educational Benefits?. *Procedia - Social and Behavioral Sciences*. 46. 5502-5508. 10.1016/j.sbspro.2012.06.465.
- Bellotti, Francesco & Berta, Riccardo & De Gloria, Alessandro. (2010). Designing Effective Serious Games: Opportunities and Challenges for Research. *International Journal of Emerging Technologies in Learning*. 5. 10.3991/ijet.v5s3.1500.
- Marsh, Tim. (2016). Slow serious games, interactions and play: Designing for positive and serious experience and reflection. *Entertainment Computing*. 14. 454-53.
- Stapleton, Andrew. (2004). Serious games: Serious opportunities. *Australian Game Developers Conference*.
- videogameeffects.weebly.com, ΚΑΤΗΓΟΡΙΕΣ ΗΛΕΚΤΡΟΝΙΚΩΝ ΠΑΙΧΝΙΔΙΩΝ
- researchgate.net, Τρισδιάστατα εκπαιδευτικά παιχνίδια, σοβαρά παιχνίδια, January 2017, Emmanuel Fokides
- Ηλεκτρονικά παιχνίδια, κίνητρα και μάθηση:Διερευνώντας το πεδίο των MMOGs, Σοφία Μυσιρλάκη, Φωτεινή Παρασκευά
- Game-based learning: αξιολόγηση του playability σε παιχνίδι τύπου Monopoly, Δορμπάρης, Δημήτριος , 2019-07
- What are Location Based Games?, Christian McCrea ,Swinburne University, 2006
- eproceedings.epublishing.ekt.g, “ Διεθνές Συνέδριο για την Ανοικτή & εξ Αποστάσεως Εκπαίδευση” ,ΜΑΚΡΗ ΑΓΟΡΙΤΣΑ & ΒΛΑΧΟΠΟΥΛΟΣ ΔΗΜΗΤΡΙΟΣ, 2017
- srv54.mednet.gr, “ Ηλεκτρονική μάθηση (e-learning) Γενική θεώρηση και εφαρμοσμένο παράδειγμα από την ειδικότητα της Ακτινοδιαγνωστικής ”, Α.Ν. Χαλαζωνίτης,Δ. Κουμαριανός,Ι. Αποστολάκης
- cv.vic.gov.au, “What are Location Based Games?”, Christian McCrea, Swinburne University, 2006
- okeanis.lib.puas.gr, ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΑΤΤΙΚΗΣ, “E-learning”, Παναγιωτοπούλου Αικατερίνη

- dione.lib.unipi.gr, ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ, ΤΜΗΜΑ ΨΗΦΙΑΚΩΝ ΣΥΣΤΗΜΑΤΩΝ, “Εκπαιδευτικά Παιχνίδια και Ηλεκτρονική Μάθηση : Διερευνώντας την κοινωνική διάσταση των πολυχρηστικών διαδικτυακών παιχνιδιών”, Σοφία Μυσirlάκη , 2010
- dione.lib.unipi.gr, ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ, “ΠΑΙΧΝΙΔΟΠΟΙΗΣΗ ΣΤΗΝ ΗΛΕΚΤΡΟΝΙΚΗ ΜΑΘΗΣΗ: ΣΧΕΔΙΑΣΜΟΣ ΣΥΣΤΗΜΑΤΟΣ ΠΑΙΧΝΙΔΟΠΟΙΗΣΗΣ ΚΑΙ ΕΦΑΡΜΟΓΗ ΤΟΥ ΣΤΗΝ ΗΛΕΚΤΡΟΝΙΚΗ ΠΛΑΤΦΟΡΜΑ MOODLE”, Δημητριάδου Σοφία, 2016
- korinthos.uop.gr, “ Ηλεκτρονικά παιχνίδια, κίνητρα και μάθηση: Διερευνώντας το πεδίο των MMOGs”, Σοφία Μυσirlάκη, Φωτεινή Παρασκευά, Τμήμα Ψηφιακών Συστημάτων, Πανεπιστήμιο Πειραιώς
- growthengineering.co.uk, “10 SERIOUS GAMES THAT CHANGED THE WORLD”, 2016
- game-learn.com, “Eight examples that explain all you need to know about serious games and game-based learning”, Gamelearn Team
- sciencedirect.com, “Serious games and learning effectiveness: The case of It’s a Deal!”, January 2012, Victoria Guillén-Nieto & Marian Aleson-Carbonell
- eludamos.org, “Urban Games to Design the Augmented City”, Vanessa De Luca, Maresa Bertolo, Eludamos. Journal for Computer Game Culture. 2012; 6 (1), pp. 71-83
- dl.acm.org, “Designing an Escape Room in the City for Public Engagement with AI-enhanced Surveillance”, Tomo Kihara, Roy Bendor, Derek Lomas, 2019
- adventure.howstuffworks.com, “How Urban Gaming Works”, ED GRABIANOWSKI

ΥΛΟΠΟΙΗΣΗ

React

- MarkupBox.com, “AngularJS vs ReactJS vs NodeJS: Which One Is best?” - Justin Spencer
- konstantinfo.com, “AngularJS vs NodeJS vs ReactJS: Which One Would You Choose?”, Helpful Resources , 16.09.2019
- reactjs.org, “Tutorial: Intro to React”
- reactjs.org, “Introducing JSX”
- reactjs.org, “Create a New React App”
- reactjs.org, “Add React to a Website”
- reactjs.org, “Rendering Elements”
- w3schools.com, “What is React?”
- fullstackreact.com, “30 days of react”, Ari Lerner
- skillcrush.com , “TECH 101: WHAT IS REACT JS?”, Scott Morris, 14.5.2019
- edureka.co , “What Is React? – Unveil The Magic Of Interactive UI With React”, Swatee Chand, Apr 29.2020
- medium.com, “A Beginner’s Guide to React with ES6”, Edmond Atto, Apr 30.2018
- c-sharpcorner.com, “What and Why React.js”, Nitin Pandit, Mar 05.2020
- simform.com, “5 Reasons to use Nodejs with React for Web Development”, Prayaag Kasundra
- medium.com, “Why ReactJS is gaining so much popularity these days”, Thinkwik, Dec 6.2017
- en.wikipedia.org, “React (web framework)”, March 19, 2020

Node.js & React

- daveceddia.com, “Do I need Node.js in the backend?”, Dave Ceddia, JUNE 08, 2016
- stackoverflow.com, “does react really need nodeJS on the frontend ENV?”, 2017

- quora.com, “Does React require Node.js?”, Luke Fabish, 2017

Node.js

- nodejs.org, “About Node.js”
- toptal.com, “Why The Hell Would I Use Node.js? A Case-by-Case Tutorial”, TOMISLAV CAPAN
- w3schools.com, “Node.js Introduction”
- en.wikipedia.org, “Node.js”, May 27, 2009
- nodejs.dev, “Run JavaScript Everywhere.”

Npm

- docs.npmjs.com, “About npm”
- npmjs.com, “Get npm!”
- w3schools.com, “What is npm?”

Npm start

- flaviocopes.com, “An introduction to the npm package manager”, Apr 17, 2019
- docs.npmjs.com, “npm-start - Start a package”

DOMS

- w3schools.com, “What is the HTML DOM?”
- w3schools.com, “JavaScript HTML DOM”
- developer.mozilla.org, “Introduction to the DOM”, MDN contributors, Aug 30.2020

React DOM

- reactjs.org/docs, “DOM Elements”
- reactjs.org/docs, “ReactDOM”
- reactjs.org/blog, “DOM Attributes in React 16”, Dan Abramov, September 08.2017

React ES6

- codeburst.io, “JavaScript — WTF is ES6, ES8, ES 2017, ECMAScript...?”, Brandon Morelli, Aug 28.2017

React Components

- reactjs.org, “Components and Props”
- techdiagonal.com, “Chapter 3 – Understanding ReactJS Components”, Suhas Holla

React Google Maps

- tomchentw.github.io, “React Google Maps Style Guide”
- dev.to, “How to use the Google Maps API with React.js”, jessicabetts, Apr 25.2019

React Bootstrap

- react-bootstrap.github.io, “React Bootstrap”
- getbootstrap.com, “Build fast, responsive sites with Bootstrap”

React Confetti

- npmjs.com, “react-confetti”, July 2020
- daniel-lundin.github.io, “React DOM Confetti”
- npmjs.com, “react-dom-confetti”

JSON

- json.org, “Introducing JSON”
- w3schools.com, “JSON – Introduction”

VSCODE

- code.visualstudio.com,
- code.visualstudio.com/docs, “Why did we build Visual Studio Code?”

Προβλήματα & Λύσεις

- upmostly.com, “Changing the Background Color in React”

KATANOHΣH

Google Maps

- developers.google.com , “Get Started with Google Maps Platform”
- medium.com, “A Beginner’s Guide to google-map-react”, Matthew Croak, Nov 10, 2017
- scotch.io, “React Apps with the Google Maps API and google-maps-react” , Rachael Njeri, October 09, 2018

API keys

- console.developers.google.com, “APIs & Services”,
- magellanic-clouds.com , “Google Cloud Platform billing settings”

If statements

- blog.logrocket.com, “8 conditional rendering methods in React”, Esteban Herrera, February 28. 2018
- stackoverflow.com, “how to compare state variables of 2 components in react?”

Maps (Ονομασίες, συγκρίσεις & κινήσεις εικονιδίων/markers)

- ej2.syncfusion.com, “Maps”
- stackoverflow.com , “How can animation be added to markers in “react-google-maps”?”
- developers.google.com , “Marker”
- sitepoint.com , “Creating Animated Google Map Markers with CSS and JavaScript”

CSS

- create-react-app.dev, “Adding a Stylesheet”
- fdossena.com, “COOL AND RESPONSIVE BUTTONS WITH A FEW LINES OF CSS”, Federico Dossena
- w3.org, “Web Style Sheets CSS tips & tricks”

Modals

- github.com/reactjs/react-modal, “any way to show an image in a modal window?”, 2016
- reactjs.org, “Handling Events”
- bootsnipp.com, ““Vertically Centered Bootstrap Modal””
- blog.bitsrc.io, “Build a simple Modal Component with React”, Eden Ella, May 15, 2019
- stackoverflow.com, “(React) Opening a Modal on Google Maps Marker Click”
- [react-bootstrap](https://react-bootstrap.com), “Modals”
- material-ui.com , “Modal”
- w3schools.com, “CSS Styling Images”
- storyblok.com , “How to render dynamic component defined in JSON using React”, Dominik Angerer ,Sep 26, 2019

onClick μέθοδο

- upmostlly.com, “Pass a Parameter Through onClick in React”, James King, October 15, 2019
- ozmoro.com, “How to pass a value to onClick event handler in React.js”, Mon Jul 02, 2018

Score (πως να κανει update την συνάρτηση)

- itnext.io, “Building a Simple Scoreboard with React”, Ethan Hansen Apr 7, 2019

Εικονίδια Marker

- flaticon.com

Δημιουργία Τίτλου

- piklr.com

Πληροφορίες Μουσείων / Ερωτήσεων / Εικόνες των Modals

- matk.gr, “ΜΟΥΣΕΙΟ ΑΣΙΑΤΙΚΗΣ ΤΕΧΝΗΣ ΚΕΡΚΥΡΑΣ”

- eksmouseiosolomou.gr, “Μουσείο Διονυσίου Σολωμού”
- alphapolitismos.gr, “ΜΟΥΣΕΙΟ ΧΑΡΤΟΝΟΜΙΣΜΑΤΩΝ ΙΟΝΙΚΗΣ ΤΡΑΠΕΖΗΣ - ΚΕΡΚΥΡΑ”
- atcorfu.com, “Το Καμπαναριό της Αβουντσίατα στην Κέρκυρα”, GasTheo, 11 NOE 2017
- projectcorfu.com, “Το καμπαναριο της Αβουντσίατα”

Ονόματα

- matk.gr, “Χατζηβασιλείου Νικόλαος”
- museumfinder.gr, “Ναπολέοντα Σαγιά”
- matk.gr, “Γρηγορίου Μάνου”
- benaki.org, “Ν. Χατζηκυριάκου-Γκίκα”

Παρουσίαση των πληροφοριών

- cubui.com, “How to render arrays in React JS”, Andy, 2018
- telerik.com, “A Beginner’s Guide to Loops in React JSX”, Thomas Findlay January 24, 2020
- reactjs.org, “Lists and Keys”
- robinwieruch.de, “How to manage React State with Arrays”, ROBIN WIERUCH, MAY 17, 2020

Goormide

- ide.goorm.io, official site
- help.goorm.io, “What is goormIDE?”

Github & GithubDesktop

- desktop.github.com, official site
- github.com, official site
- ellak.gr/wiki, “Οδηγίες χρήσης του Github”

Git

- mariosv.gr, “Λίγα λόγια για το Git”, Βασιλείου Μάριος
- git-scm.com, official site