DOSSIER DE PROJET

# Projet : Calculatrice réseau

**PLAN DU DOSSIER DE PROJET :**

1. *Présentation du cahier des charges (Dans un document Word ci-joint).*
2. *Présentation de la maquette (Incluse dans le document cahier des charges).*
3. *Liste des fonctions implémentées et leurs rôles.*
4. *Description des jeux d’essais des fonctions et résultats.*
5. *Sélection d’un bout de code avec explication.*
6. *Difficultés rencontrées.*
7. *Conclusion.*

## Liste des fonctions implémentées et leurs rôles

Les fonctions implémentées dans cette application sont les suivantes :

1. **validIp** : Valider l’affichage de l’adresse IP saisie sous la forme de 4 sockets séparés par un point et surtout vérifier la valeur de saisie entre 0 et 255.
2. **validMask** : Valider l’affichage de l’adresse Mask même processus que l’IP.
3. **classIp** : Attribuer une classe A, B, C, D ou E selon la première valeur de l’IP saisie par l’utilisateur.
4. **dec2Bin** : Prend comme valeur une adresse décimale et la convertie en binaire.
5. **bin2Dec** : Prend comme valeur une adresse binaire et la convertie en décimal.
6. **bitwiseAND** : Prend comme valeur deux adresses binaires pour faire l’addition entre 0 et 1 avec « & ».
7. **bitwiseOR** : Prend comme valeur deux adresses binaires pour faire l’addition entre 0 et 1

avec « OR ».

1. **invertedBinary** : Nous sert dans le calcul de l’adresse Broadcast pour invertir le Mask binaire et faire le calcul avec « bitwiseOR ».
2. **calcNetAddressIP** : Calcul l’adresse réseau et prend comme paramètres l’adresse IP et le Mask en utilisant la fonction « bitwiseAND ».
3. **calculateBroadcastIP** : Calcul l’adresse de diffusion (BC) et prend comme paramètres l’adresse IP et le Mask en utilisant la fonction « bitwiseOR ».
4. **firstAdd** : Prend comme paramètre l’adresse réseau comme paramètre, utilise la fonction « dec2Bin » et ajoute 1 après avec la méthode « splice ». Ensuite avec la fonction « bin2Dec » on récupère la première adresse.
5. **lastAdd** : Prend comme paramètre l’adresse BC comme paramètre, utilise la fonction « dec2Bin » et supprime 1 après la méthode « splice ». Ensuite avec la fonction « bin2Dec » on récupère la dernière adresse.
6. **nbBitMask et hostNb** : prennent en paramètre l’adresse Mask binaire, la première avec la méthode « indexOf » (prend 0 comme valeur) et on aura en return le nombre des « 0 ». Avec la deuxième on fait la soustraction par 32 (nbre total des bits) et on return on applique « Math.pow(2, résultat de la soustraction) - 2 » (- 2 pour la 1ère et dernière adresse.
7. **colMask** : prend en paramètre l’adresse Mask binaire, transforme l’adresse en couleurs « red », « green » et « **black** » pour avoir les points. On aura besoin pour donner des couleurs aux « span » et par la suite avoir le résultat attendu :



### Description des jeux d’essais des fonctions et résultats

Pour la réalisation des jeux j’ai utilisé « QUnit » afin de tester les fonctions et comparer le résultat attendu « expected » avec la valeur en entrée en utilisant « assert ».

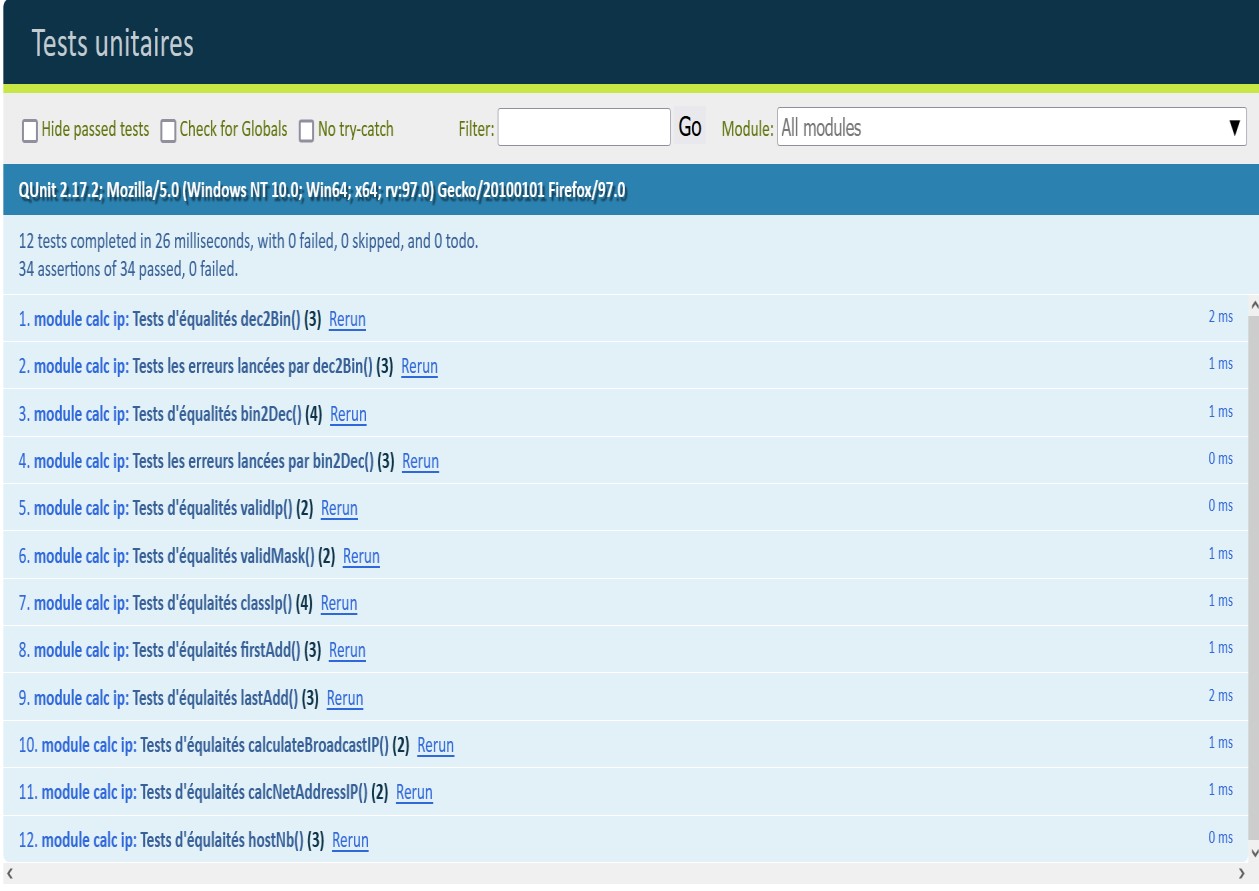
Avec l’interface de QUnit on peut vérifier la validation ou non de nos tests :

On va prendre un exemple simple comme la fonction «

firstAdd

»

:



On a bien testé les paramètres, dans ce cas des adresses réseaux, on vérifie bien si on aura en sortie des adresses avec « +1 ». Après on voie bien dans l’interface de QUnit que le test est bien validé.

### Sélection d’un bout de code avec explication

Je vais rester sur la même fonction « firstAdd » pour expliquer ce bout de code ainsi choisi :



On exporte la fonction « firstAdd » qui prend en paramètre l’adresse réseau, on converti ensuite l’adresse réseau binaire avec la fonction « dec2Bin » et on la stocke dans un tableau. Ensuite on prend la dernière valeur du tableau « [3] » comme on a 4 valeurs dans le tableau correspondant aux 4 sockets de l’adresse réseaux binaire (après l’avoir converti en nombre avec « Number ») et on ajoute « +1 ». Après on converti en string avec « toString » la valeur stockée avec la méthode « splice ».

Avec la méthode « splice » on ajoute « netAddStr » à la 3ème valeur du tableau stocké :



En utilisant la fonction « bin2Dec » on converti l’adresse réseau en binaire en ajoutant aussi la méthode « join(‘.’) » pour avoir un point entre les éléments du tableau en les collant.

A la fin on aura en return la 1ère adresse ici dans notre exemple « netAddIP ».

### Difficultés rencontrées

La plus grosse difficulté dans la réalisation de ce projet c’est surtout d’avoir les couleurs vert et rouge pour les « id. réseau » et « id. hôte ».

Travailler sur « QUnit » aussi pas évident mais après consultation et documentation j’ai pu comprendre les fonctionnalités et l’utilité de ce « unit test js framework » donc c’est un « framework JavaScript » pour les tests.

La séparation du « main.js » et « script.js » n’était pas évident pour moi au début mais avec la pratique je commence à avoir les bases pour comprendre le « DOM ».

### Conclusion

Je suis finalement satisfait de ce que j’ai acquis et j’ai pu faire dans ce projet.

Je remercie mon formateur LAMBOTIN Thomas et mes collègues qui m’ont aidé pour avancer dans la réalisation de ce projet et continuer la suite de cette formation.

