|  |
| --- |
| **JAVASCRIPT NIVEAU 2** |

|  |
| --- |
| **Var et Let** |

**Q1.**Regarde la vidéo ci-dessous et expliquez la différence entre les déclarations de variables Varlet et const :

<https://www.youtube.com/watch?v=59YAPY2WYU0>

var est accessible dans la fonction et en dehors de la fonciton

Let est accessible dans le bloc et pas en dehors de la fonction

Un objet qui est déclaré dans une constante ne va pas être modifier dans une globalité en revanche on peut modifier chaque entrée dans l’objet

**Q2.**Quelle est la variable à choisir pour déclarer une variable d’incrémentation pour une boucle for ?

La variable let est la plus utile pour la boucle for

|  |
| --- |
| **Callback** |

Une fonction de rappel (aussi appelée callback en anglais) est une fonction passée dans une autre fonction en tant qu'argument, qui est ensuite invoquée à l'intérieur de la fonction externe pour accomplir une sorte de routine ou d'action.

**Q3.**Tester et commenter le programme ci-dessous.

La fonction départ lance la fonction dite « callback » qui a pour argument affichenom , qui va en premier temps demander le nom en prompt puis envoyer ce nom dans une autre fonction pour le traduire en texte avec l’ajout de salut et -\_-

|  |  |
| --- | --- |
| **script.js** | **index.html** |
| functionafficheNom (nom){  alert('Salut ' + nom + ' -\_-')  }  functionlireNom(callback){  votreNom=prompt('Entrez votre Nom');  callback(votreNom);  }  functionDepart(){  lireNom(afficheNom);  } | <!DOCTYPEhtml>  <html>  <head>  <title>Javascript Test</title>  <scriptsrc="./script.js"></script>  </head>  <body>  <h1>Javascript Test</h1>  <p>Tester votre fonction : <buttononclick="Depart()">Test</button></p>  </body></html> |

**Q4.**Regarder la vidéo suivante tester et expliquer le code synchronique et asynchrone. <https://www.youtube.com/watch?v=XKVgCCjz9EY&t=285s>

Le code synchronique suivera l’ordre d’éxécution même si il faut attendre un timer , tous le contraire du asynchrone qui outrepassera le timer et passera a la prochaine etape et reviendra a l’ancienne etape quand le timer sera terminer

Commenter le code ci-dessous et préciser pourquoi est-il asynchrone ensuite le mettre sous forme synchronisée en remplaçant console.log("BONJOUR") par un deuxièmesetTimeout .

Ce code est asynchrone a cause du timer qu’a l’étape 1

Ref « Résynchronisation avec le setTimeout.js » pour la deuxièmes partie de la question

|  |
| --- |
| // FONCTION DE CALLBACK ------------------  constsnir =(ARG)=>{  console.log(ARG)  }  //----------------------------------------  setTimeout(snir,3000,"BTSSNIR"); //ETAPE 1 -- EXECUTE 2  console.log("BONJOUR");//ETAPE2 --EXECUTE 1 |

|  |
| --- |
| **Objets Javascript** |

Tester le code en suivant :

|  |  |
| --- | --- |
| **script.js** | **Index.html** |
| **varpersonne = {**  **prenom:"John",**  **Nom:"Doe",**  **age:50,**  **yeux:"blue"**  **};**  **functionafficher(){**  **console.log(personne.Nom);**  **console.log(personne["prenom"]);**  **}** | **<!DOCTYPEhtml>**  **<html>**  **<head>**  **<title>Javascript Test</title>**  **<scriptsrc="./script.js"></script>**  **</head>**  **<body>**  **<h1>Javascript Test</h1>**  **<p>Tester votre fonction : <buttononclick="afficher()">Test</button></p>**  **</body></html>** |

Aller sur le navigateur cliquez sur inspecter ensuite sur console pour visualiser le résultat.

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

Un objet javascript peut également prendre une fonction comme élément

tester le code suivant:

|  |
| --- |
| varpersonne = {  prenom:"Jules",  Nom:"Ferry",  age:50,  informarion:function(){  return'je suis ' +this.prenom + ' ' + this.Nom + ' ' + "j'ai " + this.age + ' ans';  }  };  functionafficher(){  console.log(personne.Nom);  console.log(personne["prenom"]);  console.log(personne.informarion())  console.log(personne);  } |

**Q5.**Expliquer pourquoi on utilise this sur les éléments suivant this.prenomthis.Nomthis.age

Pour optimiser le code on prend les info qui sont dans le même object que la fonction

Voir la vidéo

<https://www.youtube.com/watch?v=gvicrj31JOM>

|  |
| --- |
| **Les propriétés Array(Foreach,map,slice,reduce)** |

Tester et commenter le code suivant :

|  |
| --- |
| varpersonnes =[  {  prenom:"Jules",  Nom:"Ferry",  age:50,  informarion:function(){  return'je suis ' +this.prenom + ' ' + this.Nom + ' ' + "j'ai " + this.age + ' ans';  }  },  {  prenom:"gerard",  Nom:"briaud",  age:30,  informarion:function(){  return'je suis ' +this.prenom + ' ' + this.Nom + ' ' + "j'ai " + this.age + ' ans';  }  }  ];  functionafficher(){  personnes.forEach(element=>console.log(element.informarion()));  } |

**Q5.**Expliquer la ligne de code element=>console.log(element.informarion())

Voir :

C’est une fonction qui fera pour chaque x console.log(x.information) ici le x étant personne définit avant dans le personnes.foreach

<https://developer.mozilla.org/fr/docs/Web/JavaScript/Reference/Fonctions/Fonctions_fl%C3%A9ch%C3%A9es>

**Q6.**Refaire le code avec une fonction non fléchée

Refsans la foncton flecher.html dans le github

**Q7.**Visualiser la vidéo suivante pour comprendre que array en javascript c’est un objet et remplir le tableau ci-dessous avec les différentes propriétés de l’objet

<https://www.youtube.com/watch?v=oigfaZ5ApsM>

|  |  |
| --- | --- |
| Propriété | Description |
| map | Créer un nouveau tableau avec le resutalt qu’on demande comme par exemple mettre tous un tableau en x2 |
| reduce |  |
| join | Changer tous les élément du tableau en 1 chaine de caractère |
| slice | Renvoie les objet du tableau mais qui n’est pas d’origine avec des élement en moins souhaite appeler portion la portion est définie par un indice de début et un indice de fin (exclus) |
| filter | Sers a séparé ce qu’on veut de ce qu’on veut pas en filtrant avec des conditions comme la taille ou une fonciton |
| Foreach | permet d'exécuter une fonction donnée sur chaque élément du tableau. |
| push | Rajoute a la fin du tableau l’élement souhaiter |
| length | Renvoie la longueur du tableau |
| find | renvoie la **valeur** du **premier élément trouvé**dans le tableau qui respecte la condition donnée par la fonction de test passée en argument. Sinon, la valeur [undefined](https://developer.mozilla.org/fr/docs/Web/JavaScript/Reference/Global_Objects/undefined) est renvoyée. |

Ajoutez le code ci-dessous à la fonction afficher.

|  |
| --- |
| constliste= personnes.map(x=>'<li>' + x.informarion()+'</li>');  console.log(liste); |

**Q7.**Que fait la fonction mapsur le tableau ?.

Créer un tableau créant automatiquement les liste pour l’html avec la fonction informarion

**Q8.**Quelle est la forme de la variable liste ? . Tester le code ci-dessous à la place de

C’est du string

console.log(liste)

|  |
| --- |
| console.log(liste[0]);  console.log(liste[1]); |

Ajoutez le code ci-dessous à la place de

console.log(liste[0]);

console.log(liste[1]);

|  |
| --- |
| consthtml=liste.join('');  console.log(html); |

**Q8.**Quelle se passe-t-il avec la propriété join('') ?

Il transforme tous en 1 string avec pour espacement ‘ ‘

**Q9.**Pour afficher la variable html dans une page html utilisé le code ci-dessous et faire une conclusion

Avec dernière ligne ,graceal’id affiche , on affiche le résultat de la liste html dans une liste HTML

|  |
| --- |
| consthtml=liste.join('');  console.log(html);  document.getElementById("affiche").innerHTML= '<ul>'+ html + '</ul>'; |

**Q10.**Ajouter deux personnes à votre tableau comme :

|  |
| --- |
| {  prenom:"Alexandre",  Nom:"Dumas",  age:28,  informarion:function(){  return'je suis ' +this.prenom + ' ' + this.Nom + ' ' + "j'ai " + this.age + ' ans';  }  },  {  prenom:"Remi",  Nom:"Gabin",  age:25,  informarion:function(){  return'je suis ' +this.prenom + ' ' + this.Nom + ' ' + "j'ai " + this.age + ' ans';  }  } |

**Q11** Ensuite, Faire un programme qui affiche sur la page html les personnes qui ont moins de 30 ans

Utilise la propriété filter()

Ref moins de 30ans.html

|  |
| --- |
|  |

**Q12** A la fin on souhaiterait afficher la moyennes d'âges de ce personnes en utilisant la fonction suivante:

|  |
| --- |
| functionafficher(){  constage=personnes.map(x=>x=x.age);  console.log(age);  console.log(age.length)  constsomme=age.reduce((moyenne,element)=>(moyenne+element))/age.length;  console.log(somme);} |

Pour comprendre la fonction reduce il faut visualiser la vidéo suivante:

<https://www.youtube.com/watch?v=g1C40tDP0Bk>

ensuite commenter le code précédent;

On peut également écrire le code précédent en une seul ligne comme suit :

|  |
| --- |
| constage=personnes.map(x=>x=x.age).reduce((moyenne,element)=>  (moyenne+element/personnes.length),0); |

**Q13** Maintenant afficher la moyenne d'âges sur une page HTML

|  |
| --- |
|  |

|  |
| --- |
| **Promise Javascript** |

L'objet **Promise** (pour « promesse ») est utilisé pour réaliser des traitements de façon asynchrone. Une promesse représente une valeur qui peut être disponible maintenant, dans le futur voire jamais.

Visualiser la vidéos suivantes pour comprendre le principe :

<https://www.youtube.com/watch?v=RvYYCGs45L4>

Visualiser et tester le code de la vidéo ci-dessous:

<https://www.youtube.com/watch?v=SSYt7C4sCbw>

|  |  |  |
| --- | --- | --- |
| Promesse 1 Partie 1 | Promesse 2 Partie 2 | Promesse 3 Partie 3 |
| //promesse Simple  constdemarre=newPromise((resolve,reject)=>{  constisRunning =false;  if(isRunning == true){  resolve()  }else{  reject()  }  })  demarre.then(()=>{  console.log('good');  }).catch(()=>{  console.log("Oups, une erreur")  }) | // promesse avec argument dans resolve récupéré par then  constcalcul = newPromise((resolve,reject)=>{  constresult = 200\*120;  if(!result){  resolve(result)  }else{  reject(result)  }  })  calcul.then((result)=>{  console.log('good :'+result);  }).catch((result)=>{  console.log("Oups, une erreur:"+result)  }) | // promesse avec argument dans la fonction globale  constcalcul = (num1,num2)=>{  returnnewPromise((resolve,reject)=>{  constresult = num1\*num1;  if(!result){  resolve(result)  }else{  reject(result)  }  })  }  calcul(300,900).then((result)=>{  console.log('good :'+result);  }).catch((result)=>{  console.log("Oups, une erreur:"+result)  }) |

**Q14:**Modifier le code précédent promesse 3 en utilisant la méthode asyncawait

**Q15:**Les promesses en javascript sont une solution pour les problèmes du code asynchrone.

Expliquez pourquoi !

|  |
| --- |
| **Une manière plus simple AsyncAwait** |

Visualiser la vidéo ci-dessous et tester les deux méthodes de code

<https://www.youtube.com/watch?v=V_Kr9OSfDeU>

|  |  |
| --- | --- |
| **PROMISE** | **ASYNC AWAIT** |
| <script>  //-----------------Fonction1 pour faire une requete -----------------  functionmakeRequest(location){  returnnewPromise((resolve, reject)=>{  console.log(`MakingRequest to ${location}`)  if(location === 'Google'){  resolve('Google say hi')  }else {  reject(' We can only talk to Google')  }})}  //-----------------------------------------------  //----------------Fonction2 en cours de requete ----------------------  functionprocessRequest(response){  returnnewPromise((resolve,reject)=>{  console.log('Processingresponse')  resolve(`Extra Information + ${response}`)  })}  // ----------------------------------------------------------  //------------------Faire une promesse --------------------------  makeRequest('Google').then(response=>{ //Appel fonction1->  console.log('ResponseReceived')  returnprocessRequest(response) //Appel fonction2 mettre la réponse de fonction 1 dans 2  }).then(processedResponse=>{ // faire un callback et afficher le resultat final  console.log(processedResponse)  }).catch(err=>{ //Exécute en cas d'érreur  console.log(err)})  //--------------------------------------------------------------  </script> | <script>  //-----------------Fonction1 pour faire une requete -----------------  functionmakeRequest(location){  returnnewPromise((resolve, reject)=>{  console.log(`MakingRequest to ${location}`)  if(location === 'Google'){  resolve('Google say hi')  }else {  reject(' We can only talk to Google')  }})}  //-----------------------------------------------  //----------------Fonction2 en cours de requete ----------------------  functionprocessRequest(response){  returnnewPromise((resolve,reject)=>{  console.log('Processingresponse')  resolve(`Extra Information + ${response}`)  })}  // ----------------------------------------------------------  //-----------AsyncAwait -----------------------  asyncfunctiondoWork(){  try{ // Essayer la requete  constresponse=awaitmakeRequest('Google') // Appel fonction 1 et attendre la réponse  console.log('ResponseReceived')  constprocessedResponse=awaitprocessRequest(response) // Appel fonction 2 attendre la réponse  console.log(processedResponse)  } catch(err){ //en cas d'erreur  console.log(err)}}  doWork()  </script> |

|  |
| --- |
| **Pratique !!!!!!!!!!!!!!! Fetch data** |

Pour comprendre l’échange des données !

Copier et coller sur votre navigateur l’url suivante

<https://reqres.in/api/users>

**Q16:** Que représentent l’ensemble de ces données ?

Une base de donnée en JSON

Pour voir plus claire nous allons utiliser un logiciel qui affiche les données proprement

Télécharger le logiciel ci-dessous

<https://insomnia.rest/download>

Ensuite taper votre Url dans debug ensuite Query, Choisir la méthode GET et envoyer la requête avec send

|  |  |
| --- | --- |
| https://lh6.googleusercontent.com/a-4-SqNXNGixmP7ZoGeAa9GGh7dCE9raRig0x_iVbOP35b0zBGCPqHenRmaDLCyPJTDx_uuxmcC95mKhefbbHp6rSCda0FO0SUnETWW1Y4z-8R29KXeBuzKYUuRpDXEPvTs3I7zj | https://lh5.googleusercontent.com/-xg1Ay0NDwJRwW7oOVHy709o7ZIujdzwLNV-eBtM4GJ09LyqNavraEuEXxzOfCGstWeRr1QLr4S-kWsyHpXQnt2RdeABxK7q9PLqgVdQgqbHz17tB0PIGUHx-3T48tW8Dadq0VVF |

**Q17.** Combien d’utilisateurs cette requête renvoie-t-elle ?

On peut voir qu’il y a 6 utilisateur

**Q18.** Voir sur la documentation comment obtenir un seul utilisateur.

En mettant /1 à l’url on reçois l’utilisateur 1 comme ceci : https://reqres.in/api/users/1

<https://www.youtube.com/watch?v=Yp9KIcSKTNo>

Maintenant on va utiliser les connaissances vues précédemment pour traiter ce genre de données.

Ouvrir une page HTML et écrire le code ci dessous

|  |
| --- |
| <script>      letdata="";    fetch("https://reqres.in/api/users")    .then(response=>response.json())    .then(response=>console.log(JSON.stringify(response)))    .catch(error=>console.log("Erreur : " + error));    </script> |

ensuite sur la page Web ->clic droit ->inspecter->console pour visualiser les données

|  |
| --- |
| https://lh3.googleusercontent.com/Cd7ue8Q4J7X6uz3TtAIHCHLFvcKqcR-Oda_O8pew8wbcTKs3bCBgd6FI0_kStm7HbHGLq1l_kA_kTaifheKJoYE-_4w-ajdVcRFBKn8aC8sQT-Gys4IE_6_gYd80KS_jehlSzNuw |

maintenant tester le code ci-dessous

|  |
| --- |
| <script>        fetch("https://reqres.in/api/users")    .then(response=>console.log(response))    </script> |

**Q19.** Que signifie cette réponse ? Quel est le statut et la méthode de cette requête.

**Vidéo 1:**

<https://www.youtube.com/watch?v=k7LcCCApjTg&t=129s>

elle donne les information sur l’api qu’on demande , le status estfinished 200 avec la methodget

**Q20.**Quel est le contenu du body ?

Le contenu est la data

Tester le code ci-dessous.

|  |
| --- |
| <script>    fetch("https://reqres.in/api/users/1")    .then(response=>response.json())    .then(data=>console.log(data))    </script> |

**Q21**. Que contient le body de cette requête ?

Le body de cette requête contient l’utilisateur id1

**Q22.** Quelle est la forme de ces données ?

C’est une liste

**Visualiser la vidéo ci-dessous de 0 à 2:56 pour répondre aux questions.**

[**https://www.youtube.com/watch?v=sGvEqHkDyFc**](https://www.youtube.com/watch?v=sGvEqHkDyFc)

**Q23.**Tester et commenter le code ci-dessous

Le code récupère l’api , cherche la data , créer une fonction qui forme une liste avec le data.first\_name , et avec le innerhtml on peut le voir en html .

Ici une liste qui ce répète avec Je suis + datafirstname

|  |
| --- |
| **<html>**  **<body>**  **<h1>BONJOUR</h1>**  **<script>**  **varmydata=""**  **fetch("https://reqres.in/api/users")**  **.then(response=>response.json())**  **.then(response=> {**  **letdonnees=response.data**  **donnees.map(data=>{**    **varlist = document.createElement("li");**  **list.innerHTML  = ` Je suis ${data.first\_name}`;;**  **document.body.appendChild(list);**    **})**  **}).catch(error=>console.log("Erreur : " + error));**    **</script>**  **</body>**  **</html>** |

**Q24.** Modifier le code pour avoir la liste suivante

|  |
| --- |
| https://lh5.googleusercontent.com/6Lt8WRtVd9knFuCjHmASMiPyOvLpkfCL7nWZBHBwfQTASk_l3HtlxpOBhwmx0ClNzQqn809BGgCmUxSxkMFQT0IZo1kPBzfNDcM5dEhncAueEAOL0wrEVZvH_HS15fN7zuLSj1Jk |

**Q25:**Afficher les phtos des personnes comme suit :

|  |
| --- |
| https://lh6.googleusercontent.com/Sbhx7FD5YMBCRNFuBQjL2e1u4PQ8M4xy0RS1YOVzUYPJzI9UDEXswzUxlsB7KTh4rY6Dq7K2UtQ0MlYdhPe3ovO7kVtKN_nCPM2N9E2y2sXSbQPxI7l311G8lsI_nqPr6H45h-RA |

**Ressource**

<https://www.geeksforgeeks.org/how-to-create-an-image-element-dynamically-using-javascript/>

Githubfetch data script 1

**Q25.** Expliquer les différentes fonctions ci-dessous:

|  |  |
| --- | --- |
| Fonction | Description |
| document.createElement(); | La fonction créer un élément |
| document.body.appendChild(); | Rajoute un element a la liste |
| document.querySelector(); | Retourner un element de la liste |
| document.addEventListener(,); | attache une fonction à appeler chaque fois que l'évènement spécifié est envoyé à la cible. |

|  |
| --- |
| **Pratique !!!!!!!!!!!!!!! APP** |

Créer les fichiers index.html + app.js + styles.css ensuite copier chaque code dans son fichier.

|  |  |  |
| --- | --- | --- |
| HTML | CSS | JS |
| <html>      <head>            <link rel="stylesheet" href="styles.css">      </head>      <body>            <h1>PERSONNES</h1>            <div class="content">                <form id="form">                  <input type="text" name="name" placeholder="nom">                  <input type="text" name="lastname" placeholder="prenom">                  <button>Ajouter</button>              </form>                <ul id="list"></ul>            </div>              <script src="app.js"></script>      </body>  </html> | body{      background: #306EFF;      font-family: ubuntu;  }    h1{      color: #fff;      font-size: 64px;      letter-spacing: 2px;      margin-top: 80px;      text-align: center;  }    .content{      background: #fff;      max-width: 960px;      margin: 30px auto;      padding:  20px 30px;      border-radius: 10px 120px 10px 10px;      box-shadow: 1px 3px 5px rgba(0,0,0,0.1)  }    ul{      list-style-type: none;      padding: 0;  }    li{      padding: 20px;      background: #f6f6f6;      font-size: 20px;      color: #555;      position: relative;      border-bottom: 1px solid #e6e6e6;      height: 46px;  }    li:nth-child(even){      padding: 20px;      background: #f2f2f2;  }    li span{      display: block;  }    li span:nth-child(2){      font-size: 16px;      margin-top: 6px;      color: #999;  }    li div{      position: absolute;      top: 0;      right: 0px;      background: rgba(255,255,255,0.6);      width: 40px;      text-align: center;      padding: 10px 0;      font-weight:  bold;      cursor:  pointer;  }    form input{      float: left;      width: 38%;      margin: 0;      border: 0;      border-bottom: 1px solid #eee;      margin: 0 1%;      padding: 10px;      display: block;      box-sizing: border-box;      font-size: 18px;  }    form input:focus{      outline: none;      border-bottom: 3px solid #C35817;      padding-bottom: 8px;      transition: all ease 0.2s;  }    form:after{      content: '';      clear: both;      display: block;  }    button{      border: 0;      background: rgb(172, 44, 44);      border-radius: 10px;      padding: 13px;      width: 14%;      box-shadow: -1px 0px 1px rgba(0,0,0,0.1);      font-weight: bold;      font-family: ubuntu;      letter-spacing: 1px;      color: #999;  } | var personnes =[      {      id:"0",      prenom: "Jules",       Nom: "Ferry",       age: 50,  },     {      id:"1",      prenom: "gerard",       Nom: "briaud",       age: 30,   },     {      id:"2",      prenom: "Alexandre",       Nom: "Dumas",       age: 28,    },     {      id:"3",      prenom: "Remi",       Nom: "Gabin",       age: 25,   }  ];    const cafeList = document.querySelector('#list');  const form = document.querySelector('#form');    function renderCafe(doc){      console.log(doc);      let li = document.createElement('li');      let name = document.createElement('span');      let city = document.createElement('span');      let cross = document.createElement('div');        li.setAttribute('data-id', doc.id);      name.textContent = doc.prenom;      city.textContent = doc.Nom;      cross.textContent = 'x';        li.appendChild(name);      li.appendChild(city);      li.appendChild(cross);        cafeList.appendChild(li);      cross.addEventListener('click', (e) => {         alert("Comment voulez vous me supprimer")       });  }    personnes.forEach(personne=>{          renderCafe(personne)      })    form.addEventListener('submit', (e) => {      e.preventDefault();     personnes.push({         id:personnes.length-1,          Nom: form.name.value,          prenom: form.lastname.value      });        renderCafe(personnes[personnes.length-1])      form.name.value = '';      form.lastname.value = '';  }); |

**Q27.** Expliquer le rôle des lignes de code suivantes:

|  |  |
| --- | --- |
|  |  |
| li.appendChild(name); | On hiérarchise name a la liste li |
| form.addEventListener('submit', (e) => {      e.preventDefault(); | On écoute le click sur la submit |
| personnes.length-1, | Donne le nombre d’élement -1 |
| const form = document.querySelector('#form'); | Ajoute a la liste form l’id form du html |
| personnes.forEach(personne=>{          renderCafe(personne)      }) | Pour nombre de personne faire la fonction rendercafe |
| cross.addEventListener('click', (e) => {          alert("COMMENT SUPPRIMER !!!!")       }); | Ajoute l’alert comment suprimer quand on clique sur la croix |
| renderCafe(personnes[personnes.length-1])      form.name.value = '';      form.lastname.value = '';  }); | Vide le nam et lastname du form |
| personnes.push({         id:personnes.length-1,          Nom: form.name.value,          prenom: form.lastname.value      }); | Rajoute un element a la class personne avec comme id le nombre total -1 comme nom le name venant de form et le prenom de lastname de nom |

**Q27.** Commenter le code javascript.

**Q28**. Faire inspecter sur la page web, les listes dans la balise **ul** ont des identifiants comme suit:

|  |
| --- |
| https://lh6.googleusercontent.com/as9jMh8wiJd33h3-NV1r05rQymqBQb9Nqj70d3-iOE6pOGmJiiygDhXhE4vONpa-eaRvFFiDQGcIvbviXRVB5BoEtdDby71Gqor58CtGJ64GUnUeBz60t5eYt-chce2oWusPvhu6 |

Expliquer comment ont ils obtenu un identifiant (voire code).

**Q28.** Remplace la ligne alert("Comment voulez vous me supprimer")

  par  alert(e.target.parentElement)

Que se passe-t-il ?

**Q29.** Expliquer l’élément e.target

doc

<https://developer.mozilla.org/fr/docs/Web/API/Event/target>

**Q30.** Expliquer parentElement

doc

<https://www.w3schools.com/jsref/prop_node_parentelement.asp>

**Q31.** Maintenant remplacer le code alert(e.target.parentElement) par

alert(e.target.parentElement.getAttribute('data-id'))

Expliquer getAttribute

doc

<https://developer.mozilla.org/fr/docs/Web/API/Element/getAttribute>

**Q32.** Maintenant que vous avez l’identifiant de l’élément supprimer cette élément de la liste

**doc**

<https://www.w3schools.com/jsref/met_node_removechild.asp>

**Q33.**supprimer l’élément du tableau

doc1

<https://developer.mozilla.org/fr/docs/Web/JavaScript/Reference/Global_Objects/Array/splice>

doc2

<https://www.w3schools.com/jsref/jsref_splice.asp>

|  |
| --- |
| **Stockage de données !! Cloud FIREBASE** |