Test iwaco

Réaliser par : tariq jarrari

Partie I

1. Nom et prénom : TRAIQ JARRARRI
2. Nom et prénom : Casablanca
3. En recherche active
4. developpeur backend
5. A

Partie II

1. Git est de loin le système de contrôle de version le plus largement utilisé aujourd'hui.,Git est un projet open source avancé, qui est activement maintenu. À l'origine, il a été développé en 2005 par Linus Torvalds, le créateur bien connu du noyau du système d'exploitation Linux. De plus en plus de projets logiciels reposent sur Git pour le contrôle de version, y compris des projets commerciaux et en open source.
2. SASS (qui signifie Syntactically Awesome Style Sheets) est un pré-processeur pour le langage CSS.Il a été créé il y a quelques années par Hampton Catlin et Nathan Weizenbaum. Tout comme le langage LESS (qu'on a vu dans un cours précédent) il permet de générer dynamiquement du code CSS tout en offrant une syntaxe simple et un code facilement réutilisable et maintenable.
3. Composer est un logiciel gestionnaire de dépendances libre écrit en PHP. Il permet à ses utilisateurs de déclarer et d'installer les bibliothèques dont le projet principal a besoin. Le développement a débuté en avril 2011 et a donné lieu à une première version sortie le 1er mars 2012. Développé au début par Nils Adermann et Jordi Boggiano.
4. Le pattern Object Pool est un pattern permettant de stocker une quantité finie d'instances d'une classe afin de les distribuer à qui en a besoin et qui sont rendues en fin d'utilisation. Souvent, on les utilise pour gérer des threads, des connexions distantes.

5. git diff est une commande Git multi-usage qui exécute une fonction de différenciation sur des sources de données Git. Ces sources de données peuvent être des commits, des branches, des fichiers, et bien plus.

Javascript

1. Le « Cross-origin resource sharing » (CORS) ou « partage des ressources entre origines multiples » (en français, moins usité) est un mécanisme qui consiste à ajouter des en-têtes HTTP afin de permettre à un agent utilisateur d'accéder à des ressources d'un serveur situé sur une autre origine que le site courant.

2.

== est utilisé pour la comparaison entre deux variables quel que soit le type de la variable.

=== est utilisé pour une comparaison stricte entre deux variables c'est à dire que cela vérifiera le type et la valeur des deux variables, ce qui signifie qu'il vérifiera le type et comparera les deux valeurs.

3. false

4. This est un opérateur et comme tout opérateur il retourne une valeur.

CSS

1. Vous aurez remarqué que ces exemples d'éléments block et inline sont très révélateurs : les éléments block sont structurels alors que les éléments inline sont relatifs au texte. C'est une bonne façon de se rappeler qui est qui, même si au commencement cela peut vous paraître confus

2. Pourquoi voudrais-tu faire du responsive sans media query ? Il faut pas se le cacher : travailler avec les media queries n'est pas toujours évident. Cela implique pour chaque "morceau" de

votre site ou appli qui va devoir s'adapter de prévoir un ou plusieurs breakpoints lié à la taille disponible de votre viewport

3. Le seul inconvénient du display: inline-block approche est que, dans IE7 et en dessous d'un élément ne peut être affichée inline-block si elle était déjà inline par défaut. Ce que cela signifie, c'est qu'au lieu d'utiliser un élément, vous devez utiliser un élément. Ce n'est pas vraiment un énorme inconvénient parce que sémantiquement un est de diviser la page en un est juste pour un laps de temps d'une page, donc il n'y a pas une énorme différence sémantique. Un énorme avantage de display:inline-block est que quand d'autres développeurs sont le maintien de votre code à un moment plus tard, il est beaucoup plus évident que display:inline-block et text-align:right essaye de faire qu'un float:left ou float:right déclaration.

PHP

1.

```
<?php
for ($i = 1; $i <= 100; $i++) {
    if($i%3 == 0){
    echo "Dev \n";
    }
  if($i%5 == 0){
    echo "Ops \n";
    }
  if($i%3== 0 and $i%5 == 0){
    echo "DevOps \n";
    }
}
?>
```

2.

```
<?php
function isPrime($number) {
    //boucle de 2 au nombre à tester
    for ($i = 2; $i < $number; $i++) {
        //test du qutotien de la division
```

```
        if ($number % $i == 0) {
            return FALSE;
        }
    }

    //Aucun diviseur trouvé, c'est un nombre permier
    return TRUE;
}

echo isPrime(73) ? 'Prime' : 'Composite';
?>
```

3.

Convert list à map et double tous les nombres négative.

DATABASE

1.

```
SELECT count(*),date_format(create_at,"%Y-%m-%d") FROM `employee` group by

date_format(create_at,"%Y-%m-%d")
```

count(*)	date_format(create_at,"%Y-%m-%d")
28	2021-10-01
31	2021-10-02
34	2021-10-03
28	2021-10-04
27	2021-10-05
42	2021-10-06
28	2021-10-07
29	2021-10-08
39	2021-10-09
41	2021-10-10
32	2021-10-11
39	2021-10-12
30	2021-10-13
32	2021-10-14
29	2021-10-15
26	2021-10-16
41	2021-10-17
23	2021-10-18
31	2021-10-19
45	2021-10-20
40	2021-10-21
33	2021-10-22

Console de requêtes SQL

2.

SELECT * FROM `employee` where date_format(create_at,"%Y-%m-%d")="2021-10-03"

id	first_name	last_name	email	salary	create_at
11	Kori	Gissing	kgissinga@kickstarter.com	9001	2021-10-03 17:00:23
16	Claudetta	Winram	cwinramf@wsj.com	12494	2021-10-03 05:21:26
132	Dougie	Barthropp	dbarthropp3n@usda.gov	8378	2021-10-03 01:30:01
147	Nisse	Diemer	ndiemer42@pbs.org	10767	2021-10-03 16:06:39
155	Rochette	Julien	rjulien4a@japanpost.jp	7521	2021-10-03 22:54:55
210	Paquito	O' Cloney	pocloney5t@histats.com	7549	2021-10-03 04:20:26
213	Tremain	Knapp	tknapp5w@netscape.com	9216	2021-10-03 15:47:33
229	Colas	De La Salle	cdelasalle6c@mozilla.org	7416	2021-10-03 19:18:14
231	Annalee	Tuer	atuer6e@odnoklassniki.ru	10750	2021-10-03 07:21:13
238	Clemence	Sellars	csellars6l@springer.com	7317	2021-10-03 02:26:19
267	Andres	Catford	acatford7e@stumbleupon.com	12448	2021-10-03 22:42:30
338	Raynell	Delacroux	rdelacroux9d@tripod.com	12112	2021-10-03 18:03:43
343	Chevy	Robertucci	crobertucci9i@tiny.cc	8545	2021-10-03 10:51:14
346	Carmen	Dumbare	cdumbare9l@skyrock.com	5765	2021-10-03 06:56:10
355	Joey	Hyde-Chambers	jhydechambers9u@home.pl	9204	2021-10-03 01:32:31
408	Reggie	Hooban	rhoobanbb@mac.com	5164	2021-10-03 01:34:49
439	Helenka	Huelin	hhuelinc6@cdbaby.com	5975	2021-10-03 01:53:28
485	Kanya	Pennycock	kpennycockdg@google.ru	7423	2021-10-03 11:20:24
520	Beitris	Lenton	blentonef@cpanel.net	10418	2021-10-03 12:14:24
594	Marmaduke	Iacobacci	miacobaccigh@clickbank.net	5982	2021-10-03 19:12:16
734	Kippar	Kilgallen	kkilgallenkd@dion.ne.jp	7859	2021-10-03 14:03:23
745	Ellissa	Vondrak	evondrakko@prweb.com	9018	2021-10-03 02:08:16

Console de requêtes SQL

3.

SELECT * FROM `employee` WHERE email LIKE "@consonne%"

4.

SELECT lower(first_name),lower(last_name) FROM `employee`

5.

```
ALTER TABLE employee

CHANGE create_at date_creation datetime ;
```

6.

Cependant, le point qui nous permet de différencier DELETE et TRUNCATE est que DELETE est capable de supprimer des n-uplets spécifiés d'une relation, alors que la commande TRUNCATE supprime des n-uplets entiers d'une relation.

7.

```
SELECT * from employee as c where c.salary> (SELECT MIN(b.salary)+MIN(e.salar
y) FROM employee as e,employee as b WHERE e.salary!=b.salary)
```

id	first_name	last_name	email	salary	create_at
1	Cammy	Yegorkin	cyegorkin0@usatoday.com	10188	2021-10-27 08:0
2	Shae	Masham	smasham1@ftc.gov	11601	2021-10-11 18:1
9	Edithe	Radborn	eradborn8@360.cn	12534	2021-10-15 00:5
10	Darrelle	Oxtiby	doxtiby9@taobao.com	12483	2021-10-06 11:2
12	Jordan	Speedy	jspeedyb@comsenz.com	10875	2021-10-19 07:3
13	Toinette	Bradneck	tbradneckc@merriam-webster.com	12670	2021-10-25 10:1
16	Claudetta	Winram	cwinramf@wsj.com	12494	2021-10-03 05:2
18	Sansone	Comrie	scomrieh@histats.com	11449	2021-10-26 15:0
21	Dar	Vasin	dvasink@mysql.com	10383	2021-10-29 15:5
23	Dory	St Clair	dstclairm@harvard.edu	10854	2021-10-10 14:4
24	Ardeen	Vardon	avardonn@baidu.com	11460	2021-10-11 02:1
27	Ingram	Thorneley	ithorneleyq@freewebs.com	12764	2021-10-21 21:1
29	Brigit	Pinckney	bpinckneys@parallels.com	10501	2021-10-26 02:3
32	Yank	Enderwick	yenderwickv@yahoo.com	10229	2021-10-19 10:5
35	Ellery	Di Pietro	edipietroy@skype.com	11133	2021-10-20 18:2
36	Jephthah	Marushak	jmarushakz@dailymotion.com	10889	2021-10-23 19:2
40	Murielle	Yarr	myarr13@lycos.com	11980	2021-10-27 20:1
41	Clarissa	Onn	conn14@cnet.com	10952	2021-10-19 00:0
43	Clarke	Rickarsey	crickarsey16@chronoengine.com	11107	2021-10-01 11:2
53	Audry	Smalridge	asmalridge1g@engadget.com	12655	2021-10-17 05:4
55	Vachel	Gottelier	vgottelier1i@phoca.cz	10559	2021-10-20 07:0

Console de requêtes SQL

Partie III

Pratique (web developper)