TÖL306G Vefforritun

Prófdagur og tími: 09.12.2014 09:00-12:00

Prófstaður:
Aðalbygging - A069 (fjöldi: 4)
Aðalbygging - A229 (fjöldi: 1)
Aðalbygging - A230 (fjöldi: 1)
VR-2 - V138 (fjöldi: 18)
VR-2 - V147 (fjöldi: 14)
VR-2 - V152 (fjöldi: 17)
VR-2 - V155 (fjöldi: 11)
VR-2 - V156 (fjöldi: 11)
VR-2 - V258 (fjöldi: 11)
VR-2 - V261 (fjöldi: 15)

HÁSKÓLI ÍSLANDS

Iðnaðarverkfræði-, vélaverkfræði- og tölvunarfræðideild

Skriflegt próf

Skráðir til prófs: 103

Kennari:
Ólafur Sverrir Kjartansson (osk@hi.is / GSM: 6922349) Umsjónarkennari

Kennslumisseri: Haust 2014

Úrlausnir skulu merktar með nafni

Prófbók/svarblöð:
Prófbók óþörf

Hjálpargögn:
Engin leyfileg hjálpargögn

Önnur fyrirmæli:

Aðgangur að prófverkefni að loknu prófi:
Kennslusvið sendir eintak í prófasafn

Einkunnir skulu skráðar í Uglu eigi síðar en 30.12.2014.

AHUGIÐ að einhverjar úrlausnir úr fjölmennum prófum geta verið í þunnum umslögum sem auðvelt er að yfirsjást. GÓÐ VINNUREGLA er að byrja á því að opna öll umslög, telja úrlausnir og athuga hvort fjöldi stemmir við uppgefinn fjölda sem kvittað var fyrir.

Samkvæmt 60. grein Reglna fyrir Háskóla Íslands skulu einkunnir birtar í síðasta lagi tveimur vikum eftir hvert próf, nema ef desemberpróf, þá eftir þrjár vikur. Einkunnir skulu skráðar í Uglu.

HÁSKÓLI ÍSLANDS

Verkfræði- og náttúruvísindasvið

Vefforritun

TÖL306G

Lokapróf

Kennari:	Ólafur Sverrir Kjartansson
Dagur:	9. desember 2014
Klukkan:	09:00 – 12:00
Hjálpargögn:	Engin hjálpargögn

Nafn: _____

Kennitala: _____

Prófið er 14 blaðsíður.

Gangi ykkur vel!

1. Krossaspurningar, 30%

Fyrsti hluti inniheldur 15 krossaspurningar sem hver um sig gildir 2%.

Aðeins er eitt rétt svar við hverri spurningu. Ekki er dregið niður fyrir vitlaus svör, en ef merkt er við fleira en eitt svar eru engin stig gefin fyrir þá spurningu.

1.1 (2%) Hvað er CSS *shorthand*?

☐ a. Virkni sem CSS *preprocessor*ar gefa okkur aðgang að, til að skrifa styttra og skipulagðara CSS

☐ b. Skilgreining í einni yfirlýsingu fyrir mörg gildi með svipaða virkni

☐ c. Það þegar allar CSS yfirlýsingar eru skrifaðar í einni línu fyrir hverja reglu

☐ d. Ný CSS3 virkni sem skilgreinir styttri leiðir til að skrifa CSS

1.2 (2%) Hvað eru callbacks og promises?

☐ a. Aðferðir til að vinna með *ósamstilltan* (asynchronous) kóða

☐ b. Aðferðir til að vinna með *ekki-blokkandi* (non-blocking) I/O

☐ c. Aðferðir til að vinna með *strauma* (streams)

☐ d. Aðferðir til að vinna með *atburði* (events)

1.3 (2%) Hvað af eftirfarandi á við þetta HTML:

```
<p class="intro">Hello world!</p>
```

☐ a. *p* er tag með *class* attribute

☐ b. *p* er element með *class* tag

☐ c. Byrjunar og enda element er *p*, *class* er attribute og öll heildin er *p* tag

☐ d. Byrjunar og enda tag er *p*, *class* er attribute og öll heildin er *p* element

1.4 (2%) Hvað skrifast út þegar þessi JavaScript kóði er keyrður?

```
function multiplier(x) {
    return function (y) { return x * y; }
}

var y = 10;
var m = multiplier(2);

console.log(y + m(5));
```

☐ a. 20
☐ b. 60
☐ c. 25
☐ d. 52

1.5 (2%) Hvað af eftirfarandi á við SQLite

☐ a. Lítill gagnagrunnur sem útfærir aðeins lítinn hluta af SQL staðlinum, með takmarkaðar týpur og er ekki RDBMS (Relational Database Management System)
☐ b. Lítill gagnagrunnur sem útfærir mesta allan af SQL staðlinum, með dýnamískar týpur og er ekki RDBMS
☐ c. Lítill gagnagrunnur sem útfærir aðeins hluta af SQL staðlinum, með takmarkaðar týpur og er RDBMS
☐ d. Lítill gagnagrunnur sem útfærir mest allan hluta af SQL staðlinum, með dýnamískar týpur og er RDBMS

1.6 (2%) Hvað er WCAG?

☐ a. Listi af tilmælum til að gera vefi aðgengilegri, skilgreindur af W3C
☐ b. Listi af tilmælum til að gera vefi öruggari, skilgreindur af W3C
☐ c. Listi af tilmælum til að gera vefi aðgengilegri, skilgreindur af ECMA
☐ d. Listi af tilmælum til að gera vefi öruggari, skilgreindur af ECMA

1.7 (2%) Þegar við útfærum skalanlega vefi (responsive web design) þá þurfum við að:

☐ a. Skilgreina grind með föstum stærðum og nota media-queries
☐ b. Skilgreina grind með hlutfallslegum stærðum og nota media-queries
☐ c. Skilgreina box með föstum stærðum, nota JavaScript og media-queries
☐ d. Skilgreina box með hlutfallslegum stærðum, nota JavaScript og media-queries

1.8 (2%) Þegar við biðjum um eigindi á hlut í JavaScript kemur prótótýpukeðjan til sögunnar, nánar tiltekið þá

☐ a. Ef eigindi er á hlut er því skilað, ef ekki er leitað upp prótótýpu keðjuna og leitað þar, ef ekkert finnst er `undefined` skilað

☐ b. Ef eigindi er á hlut efst í prótótýpu keðjunni er því skilað, annars er leitað niður keðjuna þar til komið er að hlutnum sjálfum, ef ekkert finnst er `undefined` skilað

☐ c. Ef eigindi er á hlut er því skilað annars er `undefined` skilað

☐ d. Ef eigindi er á hlut efst í prótótýpu keðjunni er því skilað annars er `undefined` skilað

1.9 (2%) Með CSRF, Cross-Site Request Forgery, geta óprúttnir aðilar látið notanda framkvæma aðgerðir á öðrum vefjum, óafvitandi, með því að:

☐ a. nýta XSS holur

☐ b. nýta injection árásir

☐ c. útbúa faldar HTTP beiðnir

☐ d. nýta þekktar villur í hugbúnaði

1.10 (2%) Kökur eða *cookies* eru notaðar til að

☐ a. Geyma stöðu í HTTP

☐ b. Geyma gögn á forminu nafn og gildi

☐ c. Fylgjast með notanda

☐ d. Allt að ofan

1.11 (2%) HATEOAS eða Hypermedia as the engine of application state er takmörkun í REST, hún snýst um að

☐ a. Client fær allar upplýsingar frá vefþjónustu í byrjun og notar tengla til að breyta stöðum

☐ b. Client þarf engar frekari upplýsingar en þær sem hann fær í byrjun og notar tengla til að breyta stöðum og fá frekari upplýsingar

☐ c. Client fær allar upplýsingar frá vefþjónustu í byrjun

☐ d. Client þarf engar frekari upplýsingar en þær sem hann fær í byrjun

1.12 (2%) Sniðmát eða *templating* er eitthvað sem mörg framework bjóða upp, hver er hugsunin með þeim?

☐ a. Sjá um að útbúa útlit óháð virkni

☐ b. Sjá um að útbúa góðar og skýrar slóðir, *URL*

☐ c. Sjá um að setja upp stuðning við þýðingar

☐ d. Sjá um að gera vinnu með gagnagrunna einfaldari

1.13 (2%) JSON stendur fyrir JavaScript Object Notation en það er munur á JSON og JavaScript hlutum, þar er helst að nefna:

☐ a. Lyklar í JSON verða að vera strengir
☐ b. Lyklar í JSON mega vera úr lista frátekinna orða
☐ c. Gildi í JSON mega ekki vera föll
☐ d. Allt að ofan

1.14 (2%) Hvað er skrifað út ef við keyrum þennan PHP kóða?

```
$a = array("foo" => "10");

$a[] = 10;

var_dump($a["foo"] + $a[0]);
```

☐ a. `int 20`
☐ b. `string "1010"`
☐ c. `string "20"`
☐ d. Villa, ekkert skrifast út

1.15 (2%) Ef við framkvæmum aðgerð í HTTP og stöðukóðinn sem kemur til baka er með tölugildið 500 eða hærra, gefur það til kynna að:

☐ a. Aðgerð tókst, kóði gefur til kynna hvernig það nákvæmlega tókst
☐ b. Aðgerð var beint eitthvað annað, *redirected*
☐ c. Aðgerð tókst ekki og villuna má finna hjá okkur, í client
☐ d. Aðgerð tókst ekki og villuna má finna hjá vefþjón, á server

2. Forritunarspurningar, 40%

2.1 (15%) HTML & CSS

Fyrir eftirfarandi HTML bút:

```
<div class="main">
      <article style="padding-top: 0;">
            <section>
                  <h2>Donec vel urna sem</h2>
                  <p>In purus justo, sodales vitae nisi quis.</p>
                  <img src="foo.png">
                  <a href="foo.html" onclick="javascript:link();">Foo</a>
            </section>
      </article>
</div>
```

er skilgreint eftirfarandi CSS:

```
* { margin: 0; padding: 0; }
body { font: 12px/2em Roboto; }
div.main { width: 500px; }
article {
      width: 50%;
      border: 2px solid #000;
      padding: 10px;
      padding-left: 0 !important;
      padding-right: 15px;
      height: 500px;
}
section h2 { font-size: 18px; }
```

Hægt er að gagnrýna a.m.k. sex atriði við bútana að ofan, gerðu grein fyrir þeim í stuttu máli:

1.

2.

3.

4.

5.

6.

Hver er stærðin (hæð og breidd) á `article` skv. box módelinu? Sýnið útreikninga.

2.2 (15%) JavaScript

Útfærið virkni sem staðfestir að nafn og netfang sé rétt útfyllt.

Búið er að skilgreina HTML fyrir formið og draga upp grind að JavaScript virkni. jQuery er til staðar og skal notað.

2.2.1 Hvað ber almennt að hafa í huga þegar við staðfestum gögn frá notendum m.t.t. öryggis?

HTML:

```
<ul id="errors"></ul>
<form action="index.html" method="post">
      <label for="name">Nafn:</label>
      <input type="text" name="name" id="name">

      <label for="email">Netfang:</label>
      <input type="text" name="email" id="email">

      <input type="submit" value="Senda">
</form>
```

Fyllið út fyrir bókstafina A-J viðeigandi strengi, breytuheiti eða föll þ.a. virkni sé uppfyllt. Verið nákvæm og uppfyllið málfræðireglur JavaScript.

```
$(A).submit(function (e) {
      var form = $(this), valid = true;
      $(B).empty();     // hreinsa fyrri villur
      var C = form.find('#name').val();
      var email = form.find(D).val();
      if (name === '') {
            $(E).append('<li>Nafn ógilt</li>');
            valid = F;
      }
      if (G.indexOf('.') < 0 || email.H('@') < 0) {
            $('#errors').I('<li>Netfang ógilt</li>');
            valid = F;
      }
      if (!valid) {
            e.J();
      }
});
```

A =

B =

C =

D =

E =

F =

G =

H =

I =

J =

2.3 (10%) PHP

Gefinn er grunnur að klasa sem sér um að sækja gögn frá slóð ef þau eru ekki í skyndiminni (cache) og setur þau þangað. Eftir er að útfæra `Get` fallið og þá virkni í index.php sem útbýr tilvik af klasanum og sækir gögn.

- `curl_init` setur upp cURL tengingu
- `curl_exec($url)` skilar gögnum frá $url (ekki þarf að hugsa um stillingar á cURL)
  - Skilar false ef eitthvað fer úrskeiðis
- `curl_close` lokar cURL tengingu

```
<?php // cache.interface.php

interface ICache

{

      // Sækir gögn fyrir $key, skilar false ef ekkert er til
      public function get($key);

      // Setur $value fyrir $key í cache
      public function set($key, $value);

}
```

```
<?php // index.php

// $cache er útfærsla af ICache
$cache = require("cache.php");

require("fetch.class.php");

$url = "..."; // slóð sem sækja á

// útfæra svo $data innihaldi gögn frá $url með því að nota Fetch klasann









// prentum út öll gögn frá $url
var_dump($data);
```

```php
<?php // fetch.class.php

class Fetch
{
	private $cache;

	public function __construct(ICache $cache)
	{
		$this->cache = $cache;
	}


	// Fyrir: $url er slóð sem sækja skal
	// Eftir: Get skilar gögnum frá $url eða false ef eitthvað kom uppá
	//		Ef ekkert kom uppá eru gögnin geymd í $cache
	public function Get($url)
	{
		// útfæra!









	}

	// Fyrir: $url er URL
	// Eftir: Skilar lykli sem hægt er að nota í cache
	private function key($url)
	{
		/* útfærsla ekki sýnd */
	}

}
```

3. Ritgerðarspurningar/ Essay questions, 30%

Þriðji hluti inniheldur fjórar spurningar en aðeins þarf að svara þrem sem hver um sig gildir 10%. Ef öllum spurningum er svarað gilda þrjú bestu svörin.

Vandið uppbyggingu og frágang. Stutt og hnitmiðuð svör.

3.1 (10%) Hvað er HTTP caching í vefforritum? Hvar er það útfært og hvernig? Hverjir sjá um caching?

3.2 (10%) Hvað er *Node.js*? Hvað er sérstakt við hvernig það vinnur með I/O aðgerðir? Á hverju er það byggt?

3.3 (10%) Þegar notendaumsjón er útfærð í vefforritum getur margt farið úrskeiðis sem getur valdið miklum vandræðum. OWASP skilgreinir brotna auðkenningu sem númer tvö á lista yfir helstu öryggishættur. Afhverju er það? Hvaða áhrif getur það haft? Gefið dæmi.

3.4 (10%) Þú ert beðin um að setja upp vef sem birtir gögn frá vefþjónustu tengdri gömlu kerfi þar sem hver fyrirspurn tekur mjög langan tíma.
Hvernig myndir þú haga útfærslu á framenda og bakenda?
Leitast er eftir almennri lýsingu sem sýnir heildar skilning á efni.