NSWI142 Webové aplikace – Zkouškový test

Jméno: NÁPRAVNÍK 16.1.2020

U každé otázky může být více správných odpovědí, ale také nemusí být správná žádná. Zaškrtávejte pouze odpovědi, které jsou zcela pravdivé. Z čistě formálního hlediska hodnotíme správnost odpovědí vzhledem k HTML5, CSS3, ECMAScriptu verze 6, PHP 7.x a rozhraní, která jsou implementována v aktuálních verzích prohlížečů Firefox a Chrome. Na druhou stranu nečekejte záludné otázky, které by závisely na rozdílech verzí jednotlivých technologií.

Pokud není uvedeno jinak, každá otázka je hodnocena jako celá dobře, nebo celá špatně. Z testu je možné získat až 100 bodů a tyto body jsou sečteny s body ze cvičení (max 45 bodů). Celkový počet bodů určuje výslednou známku takto:

- 121 bodů a více: výborně,
- 120 – 106 bodů: velmi dobře,
- 105 – 90 bodů: dobře,
- 89 bodů a méně: neprospěl(a).

---

1. *Uvažme následující příklad HTML5 kódu. Vyberte všechna pravdivá tvrzení.* [3 body]

```
<head>
  <link rel="stylesheet" href="default.css" type="text/css" />
  <link rel="first" href="intro.html" />
  <link rel="last" href="contributions.html" />
</head>
```

☐ Atribut `rel` udává typ vztahu. Prohlížeč ale nahlásí chybu, neboť typy `first` a `last` nezná.

☒ Elementy `<link>` reprezentují vztah mezi HTML dokumentem a jinými webovými zdroji (dokumenty, obrázky apod.).

☐ Vztahy typu `first` a `last` značí první, resp. poslední HTML stránku z kolekce, jejíž součástí je i tato HTML stránka. Prohlížeč je povinen zobrazit tyto linky uživateli jako součást zobrazení HTML stránky. 3

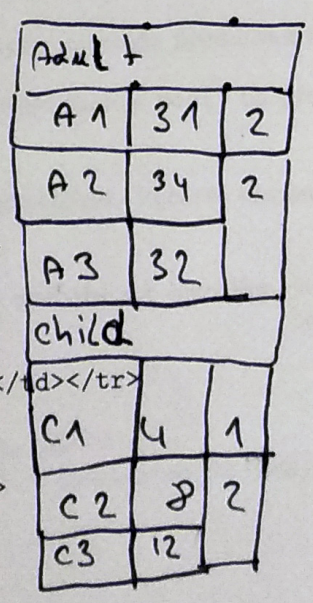
2. *Uvažme následující HTML formulář. Doplňte elementy `<input>` vhodnými atributy tak, aby je prohlížeč zobrazil uživateli jako 3 zaškrtávací pole, z nichž lze vybrat nejvýše jedno.* [4 body]

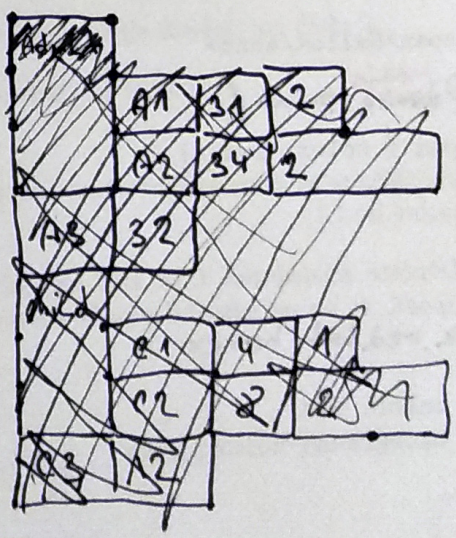
```
<form action="?">
  <p>Your age:<br>
    0-18 :  <input value="0"  type="radio" name="age" />
    19-65 : <input value="19" type="radio" name="age" />
    66-* :  <input value="66" type="radio" name="age" />
  </p>
</form>
```

4

3. *Nakreslete, jak bude následující HTML kód zobrazen v prohlížeči.* [4 body]

```
<table>
  <tr>
    <td colspan="3">Adult</td>
  </tr>
  <tr><td>A1</td><td>31</td>
    <td>2</td></tr>
  <tr><td>A2</td><td>34</td>
    <td rowspan="2">2</td></tr>
  <tr><td>A3</td><td>32</td></tr>
  <tr>
    <td colspan="3">Child</td>
  </tr>
  <tr><td>C1</td><td>4</td><td>1</td></tr>
  <tr><td>C2</td><td>8</td>
    <td rowspan="2">2</td></tr>
  <tr><td>C3</td><td>12</td></tr>
</table>
```





4

11

4. Jaký je význam/chování HTML elementu `<input>` *s atributem* `type` *nastaveným na hodnotu* `hidden`*?* [4 body]

- [x] Jedná se o pole formuláře, které není pro uživatele viditelné a uživatel ani nemůže přímo měnit jeho hodnotu, která je zapsána v HTML kódu.
- [ ] Definuje vstupní pole formuláře, které nelze měnit JavaScriptem.
- [ ] Jeho hodnota není odesílána společně s daty vyplněnými do formuláře a je tedy použitelná pouze v JavaScriptu.

4

5. Uvažme následující fragment HTML kódu. Hledáme CSS selektor, který zacílí buňku obsahující skóre hráčů, kteří nevyhráli. Vítězové jsou na řádku označeném třídou 'w'. Ostatní jsou na řádcích označených třídou s prefixem 'nw'. Skóre je ve druhém sloupci. Neuvažujeme, že bychom měnili sloupce tabulky. Mohou však libovolně přibývat nové řádky. [4 body]

```
<table>
 <tr class="nw_second"><td>John</td><td>10</td></tr>
 <tr class="nw_third"><td>Alex</td><td>9</td></tr>
 <tr class="w"><td>Thomas</td><td>12</td></tr>
 <tr class="nw_last"><td>Alois</td><td>7</td></tr>
 <tr class="nw"><td>John</td><td>8</td></tr>
 <tr class="w"><td>Frank</td><td>12</td></tr>
</table>
```

Selektor: tr[class^="nw"] td:nth-child [redacted] (2)

4

6. Která tvrzení platí pro následující CSS selektor? [4 body]

```
[class] > :nth-child(2n+1)
```

- [x] Selektor zacílí všechny elementy s atributem `class`, které se vyskytují ve svém rodiči na sudé pozici (tj. za lichým elementem).
- [ ] Selektor zacílí všechny elementy s atributem `class` s neprázdnou hodnotou, které mají lichý počet podelementů.
- [x] Selektor zacílí všechny liché podelementy (vzhledem k pozici v rámci rodičovského elementu) všech elementů s atributem `class`. Hodnota atributu `class` zde nehraje roli.
- [ ] Selektor zacílí všechny liché podelementy (vzhledem k pozici v rámci rodičovského elementu) všech elementů, které mají atribut `class` s neprázdnou hodnotu.

0

7. Doplňte následující soubor CSS pravidel tak, aby měl nápis "Hello" z následujícího HTML fragmentu červenou barvu pouze při tisku na tiskárně. [3 body]

```
<span>Hello</span>
```

```
@media device print {
span { color: red; }
}
```

3 !!

8. Doplňte následující CSS kód tak, aby se aplikoval na všechny odkazy (elementy `<a>`*) s třídou* `redtxt` *v případě, že na ně uživatel najede kurzorem.* [5 bodů]

```
a.redtxt:hover
{
  color: red;
  background-color: blue;
}
```

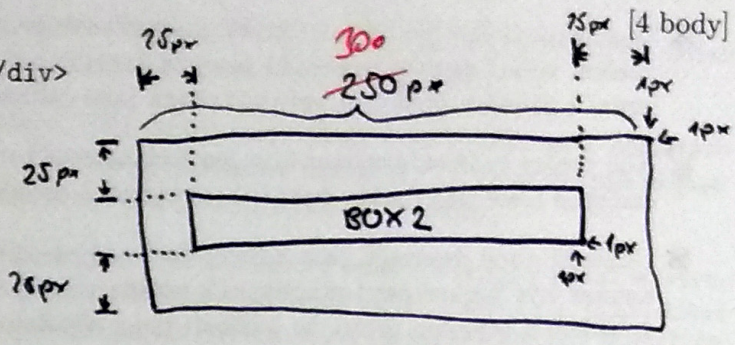
5

16

9. *Uvažme následující CSS kód aplikovaný na uvedený fragment HTML. Zakreslete, jak bude tento kód vizualizován prohlížečem (naznačte nejdůležitější rozměry v příslušných jednotkách).* [4 body]

```
<div id="box1"><div id="box2">BOX2</div></div>

#box1  {
  border: 1px solid black;
  width: 250px;
  padding: 25px;
}
#box2  {
  border: 1px solid black;
  text-align: center;
}
```



0 +

10. *Pro objekty v ECMAScriptu platí:* [2 body/odpověď]

- [x] K simulování dědičnosti používají prototypové vazby (prototype chain).
- [x] Nepoužívají tradiční koncept tříd jako běžné objektové jazyky (Java, C#).
- [x] Objekty jsou neuspořádané seznamy dvojic klíč-hodnota, kde klíč je identifikátor členské proměnné, přičemž všechny položky tohoto seznamu jsou veřejné (neexistují privátní členské proměnné).

4

11. *Uvažme následující fragment ECMAScriptu. Po vykonání tohoto skriptu bude platit:* [5 bodů]

```
function foo(a) {
  return a + 1;
}

function boo() {
  var f = foo;
  return function(a,b) { return f(a) + f(b); }
}

var fnc = boo();
foo = function(a) { return a * a; }
```

foo(2) foo(b)

fnc = (a,b) => { ret foo(2) + foo(b); }

2 → 3, 3 → 4 : 7

- [x] Výraz `fnc(2,3)` má hodnotu 7.
- [ ] Výraz `fnc(2,3)` má hodnotu 13.
- [ ] Výraz `fnc(2,3)` má hodnotu `undefined`.
- [ ] Výraz `fnc(2,3)` způsobí běhovou chybu, protože proměnná `fnc` neobsahuje funkci, ale číslo 2.
- [ ] Skript způsobí běhovou chybu při volání funkce `boo()` na předposledním řádku.

5

12. *Uvažme následující skript spuštěný v prohlížeči. Vyberte všechna pravdivá tvrzení týkající se spuštění tohoto skriptu a následného zpracování událostí.* [1 bod/odpověď]

```
<script type="text/javascript">
  function initialization() { ... }
  function periodicWork(ev) { ... }

  window.setInterval(periodicWork, 2000);
  window.setInterval(function(ev) { return periodicWork(ev); }, 3000);
  initialization();
</script>
```

9

Připomeňme, že funkce `setInterval()` nastavuje opakující se časovač a její dva parametry jsou callback funkce obsluhy časovače a interval opakování v milisekundách.

- ☒ Ze sémantického hlediska vykonávají oba časovače stejnou funkci (pouze s jiným intervalem). Explicitní obalení volání `periodicWork()` lambda funkcí (u druhého časovače) je v tomto případě funkčně ekvivalentní s přímým použitím `periodicWork` jako callback funkce.
- ☒ Kód funkcí `initialization()` a `periodicWork()` nikdy nepoběží souběžně a zároveň je zaručeno, že inicializace (celé tělo funkce `initialization()`) proběhne před prvním zavoláním funkce `periodicWork()`.
- ☒ Události obou časovačů jsou řazeny do fronty a jejich obsluhy jsou vykonávány sériově. Z toho důvodu nemusí být funkce `periodicWork()` volána přesně v daných intervalech, ale může docházet k drobným zpožděním zejména proto, že události musí nějakou dobu čekat ve frontě na obsloužení.
- ☒ Pokud by volání funkce `periodicWork()` trvalo velmi dlouho (např. jednotky vteřin), bude ovládání prohlížeče reagovat s prodlevou. V krajním případě by se mohlo dokonce stát, že dojde k zahlcení fronty událostí a prohlížeč přestane reagovat na akce uživatelského rozhraní.
- ☐ Pokud funkce `periodicWork()` zakáže při obsluze události časovače výchozí obsluhu události (vrácením hodnoty `false` nebo voláním funkce `preventDefault()`), je příslušný časovač zrušen a již nebude emitovat žádné události.

3

13. *Která tvrzení o Document Object Model (DOM) jsou pravdivá? (V této otázce se omezíme pouze na DOM tak, jak je chápán z hlediska webových technologií a HTML dokumentů.)* [1 bod/odpověď]

- ☐ DOM nepodporuje práci s kaskádovými styly, avšak vlastnosti CSS je možné zapisovat skrz DOM v textové podobě do atributů `style` jednotlivých elementů. Tyto vlastnosti jsou při zápisu do atributu zpracovány prohlížečem stejně, jako by byly přímo napsány v HTML.
- ☐ Každý tag (otevírací i zavírací) v HTML dokumentu je reprezentován jedním objektem v DOM stromě. Výjimku tvoří nepárové tagy a elementy s prázdným obsahem (kdy zavírací tag následuje těsně za otevíracím). V takovém případě není objekt zavíracího tagu přítomen.
- ☒ Všechny vnořené elementy, textový obsah, komentáře atd. daného elementu E jsou v rámci stromové hierarchie DOM uzlů potomci (děti, vnoučata, ...) objektu reprezentujícího E.
- ☒ DOM je objektovou reprezentací dokumentu, která umožnuje skriptům na straně klienta procházet a modifikovat obsah stránky.

4

14. *Uvažme modelovou situaci: Na webové stránce máme formulář, jehož úkolem je umožnit uživateli vložit novou položku do databáze. Data z formuláře jsou na straně serveru zpracována PHP skriptem. Jak musí/může být tato funkce korektně (z hlediska funkcionality, HTTP specifikace, HTML standardu, ...) implementována?* [2 body/odpověď]

- ☐ Bez ohledu na realizaci v HTML/JavaScriptu musí být příslušný HTTP požadavek řešen metodou UPDATE, jinak by skript na straně serveru neměl provádět změny v databázi.
- ☒ Jedním z možných řešení je použít asynchronní HTTP požadavek (AJAX) k odeslání formuláře. V takovém případě je nejlepší nastavit vlastní obsluhu události `onsubmit` tohoto formuláře a zakázat její výchozí obsluhu (vrácením hodnoty `false` nebo voláním funkce `preventDefault()`). Obsluha události pak může serializovat data z formuláře a odeslat je asynchronně na server.
- ☐ Bez ohledu na realizaci v HTML/JavaScriptu musí skript na straně serveru vygenerovat odpověď typu Redirect (kód 3xx), aby nedošlo k uložení této akce do historie prohlížeče a tedy aby uživatel nemohl svou akci omylem zopakovat (např. znovunačtením stránky).
- ☐ Pokud je formulář odesílán AJAXem, data v těle HTTP musí být kódována buď ve formátu XML, nebo ve formátu JSON.
- ☒ Pokud je formulář odesílán AJAXem, uživatel může během zpracování asynchronního požadavku libovolně interagovat se stránkou. Dokonce může způsobit, že prohlížeč začne načítat jinou stránku (kliknutím na odkaz, na tlačítko "zpět", ...).

10

17

15. *Jakou formou identifikuje HTTP požadovaný obsah (co má server odeslat klientovi)?* [3 body]

- [x] URI identifikující obsah může být nastaveno v hlavičce `Request-URI`. Pokud tato hlavička chybí, server automaticky vrátí výchozí dokument dle interního nastavení (zpravidla `/index.html`).
- [x] První řádek HTTP dotazu obsahuje také fragment URI, který identifikuje obsah. Dále se v některých případech používá doplňující hlavička `Host`, která obsahuje doménové jméno HTTP serveru (pro případ, že na jedné IP adrese je provozováno více webových serverů pod různými DNS jmény).
- [ ] HTTP dotaz musí obsahovat hlavičku `Content-Location`, která obsahuje relativní cestu v rámci serveru. Pokud tato cesta vede na soubor, který má být spuštěn nebo interpretován (CGI, PHP, ...), může dotaz volitelně obsahovat hlavičku `Query`, která obsahuje URL parametry pro daný skript zakódované dle MIME x-www-form-urlencoded.

0

16. *Mezi datové typy PHP patří:* [1 body/odpověď]

- [x] boolean
- [x] array
- [x] integer
- [ ] handle
- [ ] struct
- [ ] callback

5

17. *Uvažme následující PHP kód rozdělený do souborů v jednom adresáři. Jaký bude výstup ze zpracování skriptu* ***index.php****? (Bílé znaky a přesné formátování neřešte.)* [6 bodů]

first.php:

```
<p>1</p>
```

second.php:

```
<div>
  <?php include("first.php"); ?>
</div>
```

index.php:

```
<?php
  for ($i = 0; $i < 3; ++$i) {
    require("second.php");
  }
```

1

1

1

6

18. *Následující PHP kód při spuštění:* [6 bodů]

```
function bar() {
  function foo() {
    echo "foo";
  }
}
bar();
foo();
bar();
```

- [x] Způsobí běhovou chybu při prvním volání funkce `bar()`, protože PHP neumožňuje deklaraci vnořených funkcí.
- [x] Korektně vytiskne řetězec `"foo"`, ale druhé volání funkce `bar()` způsobí běhovou chybu, protože funkce `foo()` již byla deklarována.
- [ ] Způsobí běhovou chybu, protože funkce `foo()` je deklarována pouze uvnitř funkce `bar()`, takže není možné ji volat z globálního scope.
- [ ] Korektně vytiskne řetězec `"foo"`, přičemž druhé volání funkce `bar()` zbytečně znovu deklaruje funkci `foo()`.

2

11

19. *Uvažme následující formulář, který je vyplněn uživatelem a odeslán PHP skriptu na server. Předpokládejme, že uživatel pravdivě vyplní své údaje, přičemž se jedná o 16-letou bezdětnou slečnu.* [3 body/odpověď]

```
<form method="post" action="index.php">
  Age: <input type="text" name="age">
  Gender: <select name="gender">
    <option value="m"> male
    <option value="f"> female
  </select>
  Children:
    <input type="radio" name="childCount" value="0"> none
    <input type="radio" name="childCount" value="1"> one
    <input type="radio" name="childCount" value="2"> two
    <input type="radio" name="childCount" value="-1">
        three or more: <input type="text" name="childCountMore">
  <input type="submit" value="Send">
</form>
```

- Kolik položek obsahuje pole $_POST ? 4
- Jakou hodnotu jakého typu má položka $_POST['childCount'] ? string("0")
- Jakou hodnotu jakého typu má položka $_POST['childCountMore'] ? string("")

9

20. *V následujícím HTML/PHP kódu opravte všechny (potenciální) bezpečnostní chyby, které dokážete identifikovat. Stručně naznačte (1-2 věty) jak by bylo možné tyto chyby zneužít, kdyby nebyly opraveny.* [5 bodů]

cross-site scripting -> vložení <script> alert("1")</script> do jména

převedení "<" do HTML entity

```
<table>
<?php foreach($newUsers as $user) { ?>
<tr>
  escape_string 3x  htmlspecialchars()
    <td><?= $user->name ?></td>

    <td><?= $user->surname ?></td>

    <td><a ~~href="mailto:<?= $user->email ?>"~~><?= $user->email ?></a></td>
            spam boti

    <td>
        <form action="confirm.php?id=<?= $user->id ?>" method="POST">
            <button type="submit">Confirm</button>
        </form>
    </td>
</tr>
<?php } ?>
</table>
```

5 !!!

14

Celkem 78+27 bodů z maximálního počtu 100 bodů. Výsledná známka: _____.