HTML5プロフェッショナル認定試験
レベル2ポイント解説無料セミナー

株式会社クリーク・アンド・リバー社 認定講師

高井 歩

本日の内容

- 試験概要
- JavaScriptの基本

2.2.1 イベント

- イベントの発生順
- フォームイベント
- キーボードイベント
- マウスイベント
- タッチイベント
- その他のイベント
- イベントリスナ

2.2.2ドキュメントオブジェクト/DOM

- 要素の親、子要素
- 要素の表示/非表示
- 要素の上書き
- 要素の挿入/削除
- 属性の追加、取得、削除
- フォームのデータ、検証
- フォームのサブミット中止

HTML5プロフェッショナル認定資格とは

- 次世代のWebプロフェッショナルのスキルの向上に貢献するために、HTML5を活用したWebページやWebアプリケーションなどのデザイン、設計、構築に関する体系だった知識とスキルを備えたHTML5のプロフェッショナルを中立的な立場で公平かつ厳正に認定する資格制度です。

- Webデザイナー、Webプログラマー、スマートフォンアプリ開発者、サーバーサイドエンジニアなどの、Web開発プロジェクトやWebサービスに関わるあらゆるプロフェッショナルが対象です。

- 多くの企業が推進する次世代Web言語の認定資格として、HTML5のプロフェッショナルのスキルの向上に役立ちます。
また、企業内や研修機関での『技術力を担保する客観的基準』としても活用できます。

二つのレベル

HTML5 Level.1

マルチデバイスに対応したWebコンテンツを
HTML5を使ってデザイン・作成できる。

対象

- Webデザイナー
- グラフィックデザイナー
- フロントエンドプログラマー
- HTMLコーダー
- Webディレクター
- Webシステム開発者
- スマートフォンアプリ開発者
- サーバーサイドエンジニア

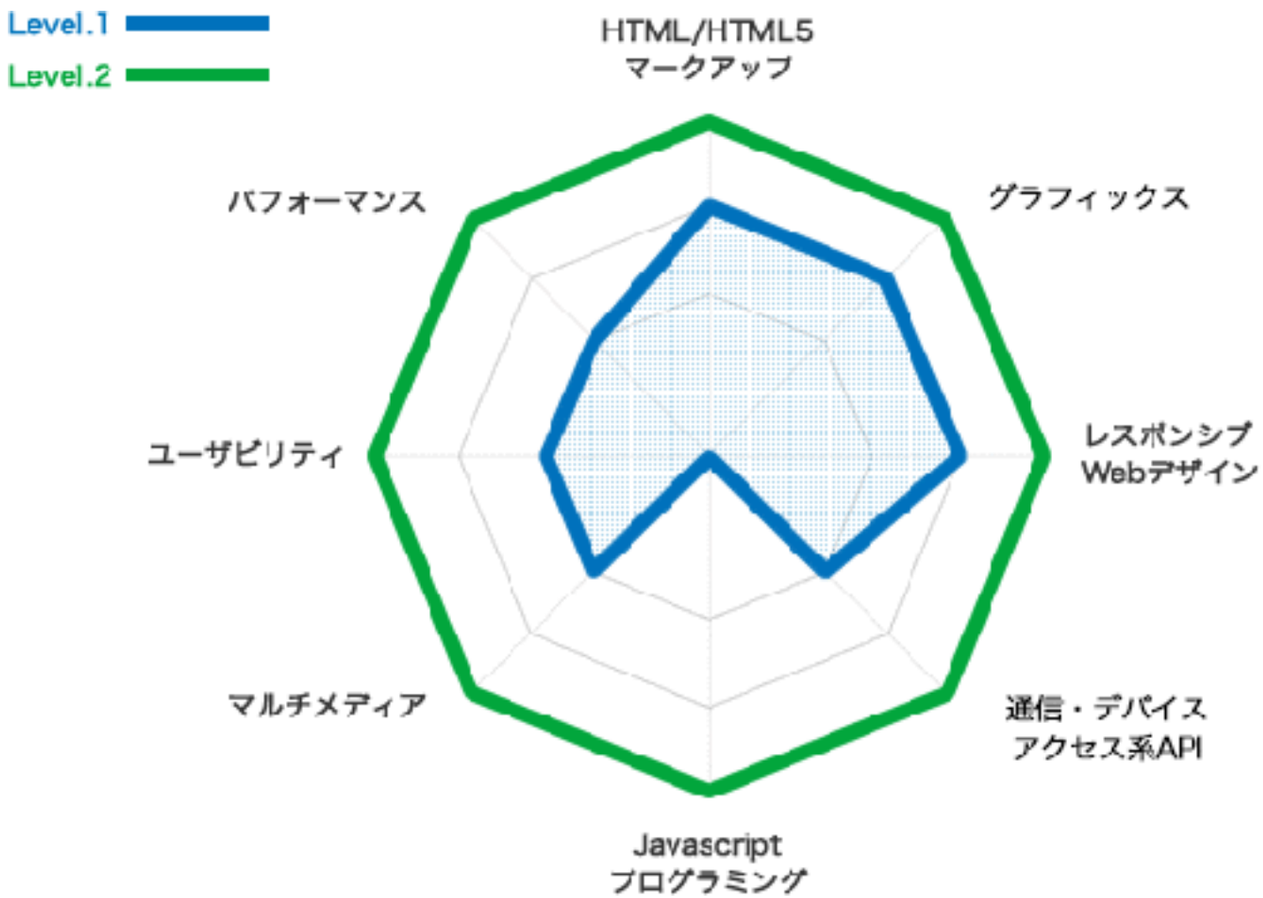
HTML5 Level.2

最新のAPIを駆使したWebアプリケーションを設計・開発できる。

対象

- Webデザイナー
- フロントエンドプログラマー
- HTMLコーダー
- Webディレクター
- Webシステム開発者
- スマートフォンアプリ開発者
- サーバーサイドエンジニア

レベル1とレベル2の資格体系



HTML/HTML5マークアップ

HTML5に関するタグの用途、構造の組み立て方に関する技術

グラフィックス

JavascriptやCSSなどを用いて、動的にグラフィックスを生成したりアニメーションを実現したりする技術

レスポンシブWebデザイン

一つのソースで、スマートフォンなどの様々なデバイスの画面サイズに対応させるための技術

通信・デバイスアクセス系API

JavaScriptからクラウドと通信をして情報の送受信を行ったり、センサーなどのデバイスにアクセスしたりする技術

Javascriptプログラミング

Javascriptを使って、動的なWebコンテンツを作成する技術

マルチメディア

３D・動画・音声ファイルなどのマルチメディアコンテンツの表示・再生に関する技術

ユーザビリティ

JavaScriptやCSSなどを用いて、デザイン仕様に沿った見やすい表示や操作しやすいコンテンツを作成するための技術

パフォーマンス

ストレージや並列処理を使ってコンテンツを効率良く高速に動作させたり、オフラインでも動作する仕組みを作るための技術

レベル1とレベル2の資格体系

ベーシックレベル
HTML5プロフェッショナル向け

所要時間：90分
試験問題数：約60問
受験料：￥15,000 （税別）
認定条件： HTML5 レベル1試験に合格すること
認定の有意性の期限：5年間

認定名： HTML5 Level1 (Markup Professional)
試験名： HTML5 Level1 Exam

この資格の認定者は、下記のスキルと知識を持つWebプロフェッショナルであることを証明できます。

- **HTML5を使ってWebコンテンツを作成することができる。**
- **ユーザー体験を考慮したWEBコンテンツを設計・作成することができる。**
- **スマートフォンや組み込み機器など、ブラウザが利用可能な様々なデバイスに対応したコンテンツを制作できる。**
- **HTML5で何ができるか、どういった技術を使うべきかの広範囲の基礎知識を有する。**

アドバンストレベル
HTML5プロフェッショナル向け

所要時間：90分
試験問題数：40〜45問
受験料：￥15,000(税別)
認定条件：HTML5 レベル2試験に合格し、かつ有意なHTML5レベル1認定を保有していること。
認定の有意性の期限：5年間

認定名： HTML5 Level2 (Application Development Professional)
試験名： HTML5 Level2 Exam

この資格の認定者は、下記のスキルと知識を持つWebプロフェッショナルであることを証明できます。

- **動的に動作させて高いユーザビリティを実現するリッチユーザインターフェイスアプリケーションを作成することができる。**
- **マルチデバイスに対応し高パフォーマンスで動作する動的コンテンツを作成することができる。**
- **システム間連携を行いリアルタイムな情報を提供するアプリケーションを作成することがでる。**
- **スマートフォンなどでネイティブアプリに近い機能を組み込んだ先端のWebアプリケーションに近い機能を組み込んだ先端のWebアプリケーションを開発することができる。**
- **APIのセキュリティモデルを理解したうえで開発することができる。**

レベル2の出題構成(1)

主題	項目	重要度
JavaScript	JavaScript文法	10
WebブラウザにおけるJavaScript API	**イベント**	**8**
	ドキュメントオブジェクト/DOM	**6**
	ウィンドウオブジェクト	8
	Selectors API	7
	History API	**7**
	テスト・デバッグ	6
グラフィックス・アニメーション	Canvas(2D)	8
	SVG	2
	Timing control for script-based animation	2
マルチメディア	メディア要素のAPI	5
ストレージ	Web Storage	7
	Indexed Database API	5
	File API	5
	バイナリーデータ	4

レベル2の出題構成(2)

主題	項目	重要度
通信	Web Socket	5
	XMLHttpRequest	5
	Server-Sent Event	1
デバイスアクセス	Geolocation API	5
	Device Orientation	1
パフォーマンスとオフライン	Web Workers	5
	High Resolution Time	2
	オフラインアプリケーションAPI	3
	Page Visibility	2
	Navigation Timing	1
セキュリティモデル	クロスオリジン制約とCORS	4
	セキュリティモデルとSSLの関係	4

学習方法

- **参考書**
- **サンプル問題**
- **出題範囲を確認**
  - 説明できない用語が無いようにする。
- **自分でサンプルを作って確かめる。**
  - 処理の順番などを確認する。(初期化->処理->後片付け)
  - Webブラウザ毎に動作が異なることがあるので注意。
  - Webサーバの有無で動作が異なることがあるので注意。

JavaScriptの基本

JavaScriptとは

- Webブラウザ内で動くプログラム言語
- 主に**オブジェクト**という部品を操作する命令を記述してプログラムを作成する。
- Webドキュメント内の要素を動かしたり、スタイルを変更させることができる
- 他にも入力フォームの入力値の検証や、サーバとの非同期通信などを行なえる
- HTML5で追加されたGPSやCanvasなどの新機能を使用するためには必須

JavaScriptの記述方法

- HTMLファイル内の<script>タグ

```
<script type="text/javascript">
  alert('サンプルプログラム');
</script>
```

type属性はHTML5で
省略可能に。

- 外部ファイルを<script>タグで読み込み

```
<script src="sample.js"></script>
```

- イベント属性に指定(イベントハンドラ)

```
<a href="#" onclick="alert('click');">スクリプト</a>
```

JavaScriptの例

```
<a href="#" onclick="clicked();">クリックして下さい</a>
<p id="msg">まだクリックされていません。</p>
<script type="javascript">
  function clicked() {
    var elm = document.getElementByld('msg');
    elm.innerHTML = 'クリックされました';
  }
</script>
```

JavaScriptの例

```
<a href="#" onclick="clicked();">クリックして下さい</a>
<p id="msg">まだクリックされていません。</p>
<script type="javascript">
  function clicked() {
    var elm = document.getElementById('msg');
    elm.innerHTML = 'クリックされました';
  }
</script>
```

a要素がクリックされたら clicked関数を呼び出し

clickedという名前の関数 (ファンクション)を定義

'msg'というid属性をもつ HTML要素のオブジェクト を取得して変数elmに代入

取得した要素の内容を 上書きする

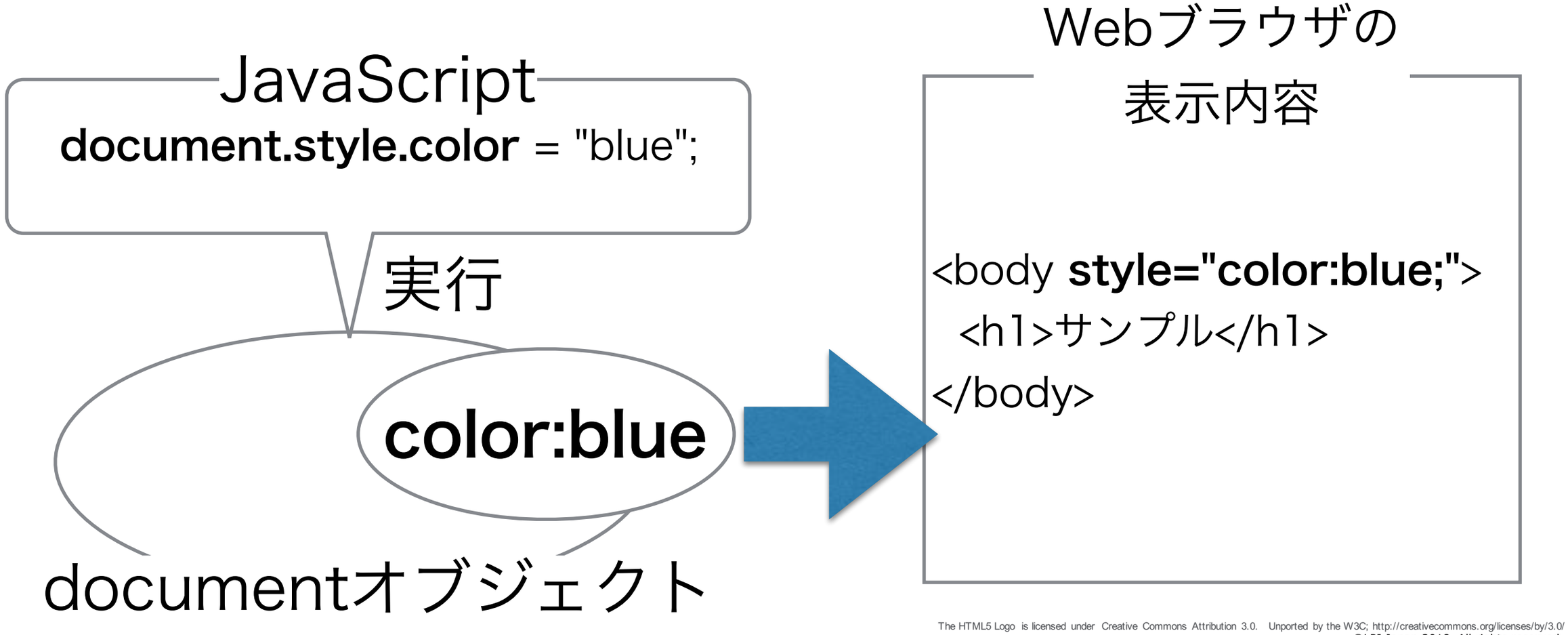
2.2.2 ドキュメントオブジェクト

DOM

知識問題 コードリーティング問題

DOM

- **D**ocument **O**bject **M**odel
- WebブラウザのJavaScriptでは、HTMLドキュメント内の**HTML要素をオブジェクトとして扱い**、操作することができる。
- ユーザがページ上で入力(操作)した内容もDOMを通じて取得できる



DOMオブジェクトの使い方

1. DOMオブジェクトを取得する
   - **var elm = document.getElementByld('sample');**
     // HTML要素のIDを指定して取得

2. DOMオブジェクトを操作する
   - メソッドを呼び出す
     - **elm.click( );** // 要素をクリックする
   - プロパティを参照する
     - alert (**elm.style.color**); // 要素に設定されている文字色を表示する
   - プロパティに値を設定する
     - **elm.style.color = 'red'**; // 要素のスタイルに指定した文字色を設定する

要素を選択する方法1

- **getElementById** … **要素のID**を指定して選択

- **getElementsByTagName** … **要素名(タグ名)**を指定して選択。結果を**コレクション**として取得。

- **getElementsByClassName** … 要素の**クラス名**を指定して選択。結果を**コレクション**として取得。

- **getElementsByName** … 要素の**name属性**を指定して選択。結果を**コレクション**として取得。

要素を選択する方法2

HTML5で追加されたAPI (出題範囲2.2.4 Selectors API)

- **querySelector … CSSと同様のセレクタ**を指定して選択。**該当する文書内の先頭**の要素を取得。

- **querySelectorAll … CSSと同様のセレクタ**を指定して選択。結果を**コレクション**として取得。

- querySelectorの例

```
var elm = document.querySelector('#test p.sample a');
```

DOMオブジェクトのコレクション

- getElementByIld,querySelector以外の方法で要素を取得した場合、結果が複数になる場合は当然のこと、0個または1個の場合でも**コレクション**になる

- コレクション内の要素に対してはまとめてメソッドを呼び出すことができないためループ文などを使用する

- コレクションに対する操作の例

```
var elms = document.getElementsByTagName('li');
for(var i = 0; i < elms.length ; i++) {
    elms[i].style.color = 'red';
}
```

要素の親、子要素

- ドキュメント全体から要素を取得する
  document.getElementByld('sample');
  documentオブジェクトに対してgetElement*メソッドを呼び出す

- 取得したDOMオブジェクトの親の要素を取得する
  var elm = document.getElementByld('sample');
  var parent = elm**.parent**;

- 取得したDOMオブジェクトの子の要素を取得する
  var elm = document.getElementByld('sample');
  var children = elm**.children**;

- DOMオブジェクトの子孫要素を条件を指定して選択することもできる
  var elm = document.getElementByld('sample');
  elm.getElementsByTagName('li');

問題1

以下のJavaScriptを実行した際に、変数testsに含まれる要素として正しいものを選択しなさい。

HTML

```
<div>
<p id="elm"><span class="test">span.test</span></p>
<a href="#" class="test">p.test</a>
</div>
```

JavaScript

```
var elm = document.getElementById('elm');
var tests = elm.parent.getElementsByClassName('test');
```

A. span要素とp要素
B. span要素
C. a要素
D. span要素とa要素

要素の表示/非表示

- displayプロパティを使う方法 (elmは対象の要素のオブジェクト)
  - 非表示にする

    elm.**style.display = 'none';**
  - 表示する (displayプロパティの値を"none"以外にする)

    elm.**style.display = 'block';**

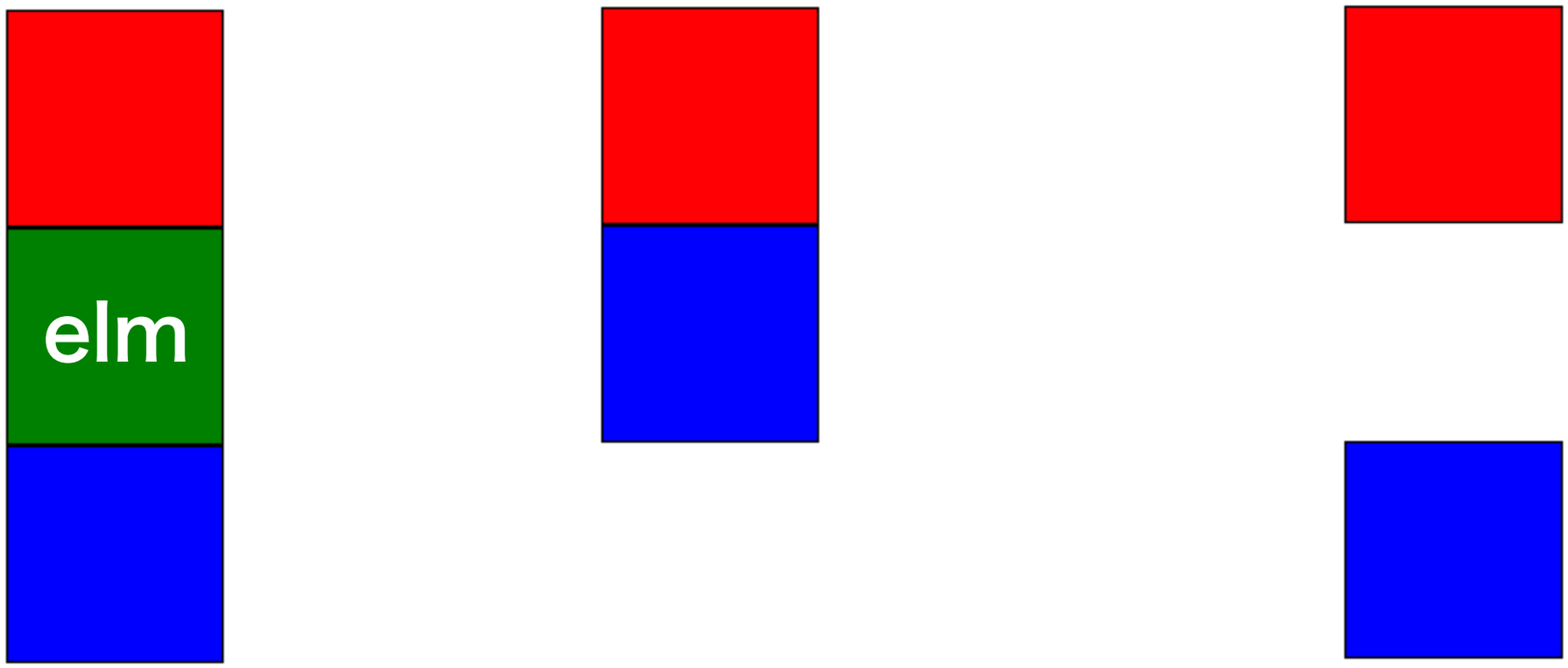
- visibilityプロパティを使う方法
  - 非表示にする

    elm.**style.visibility = 'hidden';**
  - 表示する

    elm.**style.visibility = 'visible';**

displayとvisibilityの例

- display='none'は要素のスペースごと非表示にするため、他の要素のレイアウトに影響を与える
- visibility='hidden' は要素のスペースは残し、表示だけ見えないようにする



初期状態　elm.style.display='none'　elm.style.visibility='hidden'

要素の上書き

- HTML要素のコンテンツ部分を上書きするには、**innerHTMLプロパティ**を使用する

開始タグと終了タグの間がinnerHTML
タグまで含むouterHTMLもある

- innerHTMLの例

```
<p id="sample">ここが上書きされる</p>
<script type="text/javascript">
var elm = document.getElementById('sample');
elm.innerHTML = '上書き';
</script>
```

※上書きする内容がユーザが入力したものならtextContentプロパティを使用するほうが安全。
タグやスクリプトを動作しない形式に自動的に変換してくれる。

要素の挿入/削除

- HTML要素を新しく生成しHTMLドキュメントに追加、または削除するには以下のメソッドを使用する

メソッド	概要
createElement	新しくHTML要素を生成する
appendChild	DOMオブジェクトを指定された要素の子の最後に追加する
insertBefore	DOMオブジェクトを指定された要素の子の指定された位置に挿入する
removeChild	親要素から指定された子要素を削除する

要素の挿入/削除の例

```
<ul id="list">
<li id="remove">remove me.</li>
</ul>
<script type="text/javascript">

// li要素を生成<li>item1</li>
var li1 = document.createElement('li');
li1.appendChild(document.createTextNode('item1'));

// li要素を生成<li>item2</li>
var li2 = document.createElement('li');
li2.appendChild(document.createTextNode('item2'));

var list = document.getElementById('list');
list.appendChild( li1 ); // ulの子にli1を追加
list.insertBefore( li2 , li1 ); // li1の前にli2を挿入

var li3 = document.getElementById('remove');
list.removeChild(li3); // 親要素から子要素を削除する
</script>
```

結果

- item2
- item1

属性の追加、取得、削除

- HTML要素のオブジェクトに対し、属性の追加、削除と値の取得が行なえる

- id属性やclass属性などを変更した場合、変更後に該当するスタイルがあれば適用される

メソッド	概要
createAttribute	新しく属性を生成する
setAttribute	属性をオブジェクトに追加する
getAttribute	指定した属性の値を取得する
hasAttribute	対象が指定した属性を持つか判別
removeAttribute	対象から指定した属性を削除する

属性の追加、取得、削除の例

```
<div id="target" style="color:red;">Target</div>
<script type="text/javascript">
var target = document.getElementById('target');
// 属性の生成
var attr = document.createAttribute('class');
// 属性値の設定
attr.value = 'blue';
// 属性の追加
target.setAttributeNode(attr);
// 属性の取得
console.log(target.getAttribute('class')); // blue
// 属性の削除
target.removeAttribute('style');
// 属性の有無の確認
console.log(target.hasAttribute('style')); // false
</script>
```

問題2

以下のスクリプトを実行したところエラーが発生した。エラーが発生した行はどれか。必要なHTML要素は存在するものとする。

```
var target = document.getElementById('target');
var attr = target.createAttribute('class');
attr.value = 'blue';
target.setAttributeNode(attr);
```

A. `var target = document.getElementById('target');`
B. `var attr = target.createAttribute('class');`
C. `attr.value = 'blue';`
D. `target.setAttributeNode(attr);`

フォームの選択

- HTMLドキュメント中のform要素は、他の要素と同じく、geteElement系メソッドの他に、document.forms[n]　という記述でも取得できる(※nはform要素の登場順。最初のform要素なら0)

- または、form要素にname属性がある場合は、**document.forms.属性値** もしくは**document.forms[属性値]**という記述で取得することもできる

- フォームの選択例

```
<form name="contact">
</form>
<script type="text/javascript">
// 以下の結果は同じ
var f1 = document.forms[0];
var f2 = document.forms.contact;
</script>
```

フォームのデータへのアクセス

- form要素内の入力項目の値にアクセスするには、formオブジェクトに対して、各入力項目のname属性の値のプロパティにアクセスする。

- フォームのデータへのアクセスの例

```
<form name="contact">
  <input type="email" name="mail">
</form>
<script type="text/javascript">
document.forms.contact.mail.value = "sample";
</script>
```

フォームデータの検証

- HTML5ではrequired,patternなどの属性による検証も出来るがカスタマイズ性が乏しい
- HTML5では、JavaScriptを使って簡単にフォームの入力値を検証することができる(制約検証 API)
- 要素の**validity**プロパティに入力値の検証結果がセットされている
- **setCustomValidity**メソッドで独自エラーメッセージを設定できる
- 検証の例

```
form.addEventListener('submit',function(event){
  if( email.validity.valid == false) {
    email.setCustomValidity('正しいアドレスを入力してください');
  }eles{
    email.setCustomValidity(''); // エラーメッセージをクリア
  }
});
```

サブミットの中止

- フォームに対してonSubmitイベントのイベントリスナを設定することで、フォームのSubmitを中断することができる

- サブミットの中止の例

```
document.forms[0].addEventListener('submit',function(event){
  event.preventDefault(); // デフォルト動作を中止させるメソッド
});
```

- addEventListenerを使用する場合、return false;で中止する手法は使えない

問題3

HTML中に以下の2つのフォームがあるとき、問い合わせフォームを取得しないスクリプトを選びなさい

```
<form name="search" id="search">
サイト検索
</form>
<form name="contact" id="contact">
問い合わせフォーム
</form>
```

A. document.forms.contact
B. document.forms[0]
C. document.forms[1]
D. document.getElementByld('contact')

2.2.1 イベント

知識問題 コードリーティング問題 記述問題

イベントとは

- JavaScriptの実行の切っ掛けになるもの
- イベントが発生すると、事前に設定していた JavaScriptのプログラムを実行させることができる
- マウスでボタンがクリックされた、キー入力された、スマートフォンの画面がタッチされた、などユーザーの操作によるイベントの他、HTMLファイルが読み込まれて表示の準備が出来たというようなイベントも用意されている

イベントリスナ

- HTML要素でイベントが発生した際に、プログラムが呼び出される仕組みのひとつを**イベントリスナ**と言う

- ひとつのイベントに複数のイベントリスナを同時に設定できる

- イベントリスナは追加 (**addEventListener**メソッド),削除 (**removeEventListener**メソッド)をプログラムから行なう

- 追加したイベントリスナをクリックなどのユーザ操作を必要とせずに**dispatchEvent**メソッドでプログラムから実行させることができる

イベントリスナの例

```
// イベントリスナを設定するHTML要素のオブジェクトを取得
var target = document.getElementById('target');

// イベントリスナから呼び出される関数
function clicked(event){
  console.log('click');
}

// オブジェクトにイベントリスナを追加
target.addEventListener('click',clicked);

// イベントを強制的に発生させる
target.dispatchEvent(new Event('click'));

// オブジェクトからイベントリスナを削除
target.removeEventListener('click',clicked);
```

イベント名

関数

新しいclickイベントを作る

イベント名

関数

イベントの発生順

- WebブラウザがHTMLファイルを読み込む際にもイベントが発生する

- **DOMContentLoaded** … HTMLの読み込みと解析が完了したら発生する。画像やスタイルシートなどの読み込みは待たない

- **load** … HTML解析後に画像などのリソースが読み込み終わったら発生する

- **DOMContentLoadedの後、load**イベントが発生する

問題4

以下のa要素がクリックされたらスクリプトを実行したい。正しい記述を選択しなさい。

```
<a href="#" id="btn">ボタン</a>
```

A.

```
document.getElementByld('btn').addEventListener('click',function(event){
  /* スクリプトを記述 */
});
```

B.

```
document.getElementById('a').addEventListener('click',function(event){
  /* スクリプトを記述 */
});
```

C.

```
document.getElementById('btn').addEventListener('onclick',function(event){
 /* スクリプトを記述 */
});
```

マウスイベント

- マウスイベントでは、マウスの位置、ボタンの情報などを取得できる

イベント	概要
onclick	要素がクリックされた
ondblclick	要素がダブルクリックされた
onmousedown	要素の上でマウスのボタンが押された
onmousemove	要素の上でマウスカーソルが移動した
onmouseout	要素の上からマウスカーソルが外れた
onmouseover	要素の上にマウスカーソルが入った
onmouseup	要素の上でマウスのボタンが離された
onmousewheel	要素の上でマウスのホイールが操作された
onscroll	要素がスクロールした

マウスイベントの例

```
var target = document.getElementById('target');

// 要素がクリックされたら実行される
target.addEventListener('click',function(event){

  // ローカル座標でのマウスカーソルの位置
  console.log(event.clientX+':'+event.clientY);
  // ドキュメント全体でのマウスカーソルの位置
  console.log(event.pageX+':'+event.pageY);

});
```

その他にもボタンや修飾キーの状態なども取得できる

キーボードイベント

- キーボードイベントはinput要素やtextarea要素以外にも設定できる

- キーを押して離した場合、**onkeydown**, **onkeypress**, **onkeyup**の順に発生する

- どのキーが押されたかはKeyboardEventオブジェクトの**keyプロパティ**で取得できる

- 修飾キーの状態は、**metaKeyプロパティ**、**shiftKeyプロパティ**、**ctrlKeyプロパティ**から取得できる

フォームイベント

- form要素とinput,textarea,select要素に関するイベント

イベント	概要
onfocus	要素が入力を受けとる状態(フォーカス)になった
onblur	要素がフォーカスを失なった
onchange	input,textarea,select要素の値が変化した
oninput	テキストフィールドに入力があった
onselect	テキストフィールドのテキストが選択された
onformchange	フォームの内容が変更された(現行規格では削除)
onforminput	フォームの内容が入力された(現行規格では削除)
onsubmit	送信ボタン(type="submit")が押された
oninvalid	入力欄に不正な値があり送信に失敗した
oncontextmenu	マウスの右ボタンが押された

問題5

マウスの左ボタンを押した際に発生するイベントとして正しいものを選択しなさい。

A. onmouseup

B. onmouseleft

C. onmousedown

D. onclick

ドラッグ&ドロップイベント

ドラッグされる要素のイベント

イベント	概要
ondragstart	ドラッグ開始
ondrag	ドラッグ中
ondragend	ドラッグ終了

ドロップされる領域の要素のイベント

イベント	概要
ondragenter	ドラッグしている要素がドロップ領域に入った
ondragleave	ドラッグしている要素がドロップ領域から出た
ondragover	ドラッグしている要素がドロップ領域にある
ondrop	ドラッグしている要素がドロップ領域にドロップされた

ドラッグ&ドロップの例1

```
//  ドラッグされる要素に関するイベント

var box = document.getElementById('box');

box.addEventListener('dragstart',function(event){
  console.log('dragstart');
  event.dataTransfer.setData('text/plain', 'データ');
});

box.addEventListener('drag',function(event){
  console.log('drag');
});

box.addEventListener('dragend',function(event){
  console.log('dragend');
});
```

ドラッグ&ドロップの例2

```
//  ドロップされる領域に関するイベント

var area = document.getElementById('droparea');

area.addEventListener('dragenter',function(event){
  console.log('dragenter');
});
area.addEventListener('dragover',function(event){
  event.preventDefault(); // 後の処理をキャンセルしないとondropが発生しない
  console.log('dragover');
});
area.addEventListener('dragleave',function(event){
  console.log('dragleave');
});
area.addEventListener('drop',function(event){
  event.preventDefault(); // 後の処理をキャンセルしないと別ページに移動する
  console.log('drop');
});
```

タッチイベント

- スマートフォン、タブレットやタッチ対応PCなどでは、タッチイベントを使用することができる

- マウスイベントと異なり、マルチタッチ対応のため、複数のタッチ情報を扱う

イベント	概要
touchstart	指が画面にタッチした(開始)
touchmove	タッチしている場所の移動
touchend	指が離れた(終了)

タッチイベントの例

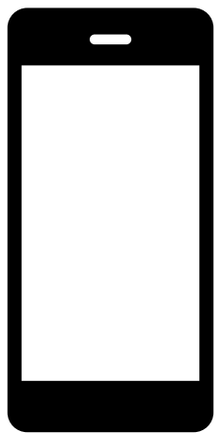
```
// ひとつめのタッチ位置のみを扱う例
window.addEventListener('touchstart',function(event){
  var touch = event.changedTouches[0];
  console.log('start'+touch.pageX+':'+touch.pageY);
});
window.addEventListener('touchmove',function(event){
  var touch = event.changedTouches[0];
  console.log('move'+touch.pageX+':'+touch.pageY);
});
window.addEventListener('touchend',function(event){
  console.log('end');
});
```

マウスイベントとタッチイベント

- マウスイベントとタッチイベントの両方に対応する場合、イベント発生の順番が問題になる
  1. touchstart
  2. touchmove(移動していれば)
  3. touchend
  4. mousemove(移動していれば)
  5. mousedown
  6. mouseup
  7. click
- **touchイベントが動作中にキャンセルされた場合、続くmosueイベントおよびclickイベントは発生しない**

スマートフォンの回転イベント

- Screen Orientation API(2018.10.12時点でドラフト)によってデバイスの向きを取得できる
- デバイスの向きが変化した際に**screen.orientation.onchange**イベントが発生する
- **screen.orientation.type**で向きを表わす文字列、**screen.orientation.angle**で角度を取得できる



type	portrait-primary	landscape-primary	landscape-primary
angle	0	90	270

カスタムイベント

- **カスタムイベント**を使うとイベントを自分で定義することができる

- 通常の関数呼び出しと異なり、イベント処理として管理を一元化できる(ひとつのイベントに簡単に複数のイベントリスナを設定できるなど)

- var event = **new Event('custom')**;
  elm.addEventListener('**custom**',function( ){ alert('custom'); });
  elm.**dispatchEvent(event)**; // イベントを発生させる

問題6

ひとつの操作に対するイベントの発生順として正しいものを選択しなさい

A. mousedown - touchstart - click

B. click - mouseup - touchend

C. touchend - mouseup - click

D. mouseup - mousedown - click

本日の内容

- 試験概要
- JavaScriptの基本

2.2.1 イベント

- イベントの発生順
- フォームイベント
- キーボードイベント
- マウスイベント
- タッチイベント
- その他のイベント
- イベントリスナ

2.2.2ドキュメントオブジェクト/DOM

- 要素の親、子要素
- 要素の表示/非表示
- 要素の上書き
- 要素の挿入/削除
- 属性の追加、取得、削除
- フォームのデータ、検証
- フォームのサブミットの中止

問題の解答

- 問題1: **D** / idが"elm"であるp要素の親の子孫のうち、classが"test"の要素はspanとa。

- 問題2: **B** / createAttributeメソッドやcreateElementメソッドはdocumentオブジェクトにしかない。

- 問題3: **B** / フォームの取得は登場順にインデックスを0から数える。

- 問題4: **A** / getElementByIdには'btn',イベント名はonを取り払った'click'を指定する。

- 問題5: **C** / ボタンの左右に係わらず、onmousedownイベントが発生する。ounmouseup,clickはボタンを離したときに発生する。

- 問題6: **C** / touch系、mouse系、clickの順で発生する。Dはdown,upの順が正しい。