# 情報理工学実験「モバイル Web アプリケーション開発」
# 実習書 2021/10/27 版

実施時期：3 年 3Q の第 3 テーマ
2021/10/29(金), 11/01(月), 11/02(火), 11/05(金), 11/08(月), 11/09(火)
実施回数：2 週で 6 回（1 回 2 コマ，週 3 回）
担当教員：飯田勝吉（情報基盤センター）
今年度はオンライン形式（Zoom 利用）で実施。

実施形式（概要）
1．自宅 PC に指定ブラウザをインストール
2．自宅 PC で教員が提供するサンプルプログラムをもとに課題を実施
3．できたプログラムを指定フォルダ（課題一時提出）にアップロード
4．教員／TA がアップロードされたプログラムをチェックし、問題があれば当該学生にそれを指摘する
5．学生は、質問がある場合に、Zoom のチャット、音声で質問する。さらに slack を提供するので、slack で質問してもよい
6．Zoom 上では、ブレイクアウトセッションの機能を使い、個別指導が可能な体制をとる（教員 1 名＋TA 2 名）ブレイクアウトに自由に出入りするため、Zoom を最新版にアップデートしておくこと。
7．最終的なレポートを ELMS で提出

指定ブラウザ：Mozilla Firefox 38.8.0 ESR
Windows 版: https://ftp.mozilla.org/pub/firefox/releases/38.8.0esr/win32/ja/
MacOS 版: https://ftp.mozilla.org/pub/firefox/releases/38.8.0esr/mac/ja-JP-mac/
Linux 版: https://ftp.mozilla.org/pub/firefox/releases/38.8.0esr/linux-x86_64/ja/

注：Windows 版以外でプログラム開発が可能か、事前に検証できていないので、Windows を利用可能な場合は、Windows で利用することを推奨する。

なお、Firefox の利用者は、複数の Firefox バージョンを共存利用する方法があるので、参考にすること。
【Win10 対応版】異なるエディション（通常/Beta/ESR）の Firefox を共存する方法
https://blog.halpas.com/archives/13257

Mac で Firefox を２つ共存させる
https://tokunosuke.net/%E3%80%90%E6%84%9F%E6%BF%80%E3%80%91mac%E3%81%A7firefox%E3%82%92%EF%BC%92%E3%81%A4%E5%85%B1%E5%AD%98%E3%81%95%E3%81%9B%E3%82%8B/

以下、演習で利用する URL などを記載する。

(*) 演習で利用するファイル群提供 URL:
https://enpit01.iic.hokudai.ac.jp/nextcloud/s/G3kCZSTKXQbRiBp

(**) 課題一時提出 URL:
https://enpit01.iic.hokudai.ac.jp/nextcloud/s/ArfYZP9Jc52W9Hj

slack の招待 URL
https://join.slack.com/t/mwa-2021/shared_invite/zt-x8qqiub8-UQpuS07xkS97SzG91EWCnw
（「メールで続行する」をクリックし、アカウントを発行すること）

(***) 一時提出プログラムの動作確認状況確認ウェブサイト URL
https://docs.google.com/spreadsheets/d/1e1kan3oHI9OhLZPcHdXQM2WwVp7ururDw8FXDLVubKs/edit#gid=0
（招待者のみがアクセスできる。受講生の ELMS のアカウントを閲覧者として登録済み）

学生の事前準備
１．指定ブラウザのインストール
２．演習ファイルのダウンロード
３．Slack の加入手続き
４．Zoom の最新版へのアップデート（2021/09/09 現在 5.7.7）

**内容概要**：
JavaScript ライブラリの一つである jQuery や jQuery Mobile などを用いて，プラットフォーム非依存のスマートフォン・タブレット向け Web アプリケーションの基礎的なプログラムを作成し，ブラウザ上で動作するクライアントサイドスクリプト言語の基本を体験的に学ぶ。

**実験環境**：
○プログラム開発・実行環境は自宅 PC とし、OS は Windows, MacOS, Linux（Windows

推奨）を利用する
○プログラムの開発には自由なエディタを使ってよい
○一部のプログラムで簡易ウェブサーバが必要になるので、指示に従い簡易ウェブサーバを利用してプログラム開発を行う（ポート番号は 8080 番とする）
○指定ブラウザ（Firefox ESR）からローカルファイルまたはローカルの Web サーバをアクセスし、プログラムの実行を確認する
○サンプルプログラムや必要なコンテンツは演習ファイル提供 URL から各自ダウンロードする
○デバッグには Firefox 標準搭載の開発ツールを使用する
○JavaScript ライブラリ（jQuery，jQuery Mobile）はコード内で URL を指定し導入
（○各自の BYOD 端末（スマホやタブレット）のブラウザから実験室内の Wi-Fi アクセスポイント経由でそれぞれのローカル Web サーバにアクセス可能とする。（本項目は、今年度実施せず。））

実験の進め方：
○テンプレートとなるサンプルプログラムをもとにして，必要な追加・変更を加え，目的の動作・機能を実現するプログラムを完成させること
○設定された到達点（実現したい動作・機能）をチェックし，実装の達成度合いを判定する
○作成したプログラムのソースと，その動作や機能がわかる実行画面のスクリーンショット等をレポートに添付すること
○プログラム開発で未解決の問題点があれば，その現象と推定される理由を考察すること
○ハッカー予備軍向けに，自由選択で取り組める発展的な部分課題にも挑戦してほしい

レポート：
○全６回分をまとめて１通のレポートを提出すること
○〆切：１１月１６日（火）２３：５９（ＪＳＴ）
○提出方法：ELMS 提出

課題一時提出について：
課題 1-1 などの課題プログラムが出来上がった場合、課題一時提出 URL (p.2, (**))にファイルをアップロードすること。教員/TA がプログラムを確認し、問題がある場合に Zoom, slack などで問題点の指摘、解決方法の議論などを行う。

一時提出を求める課題一覧：
課題 1-1，課題 1-2，課題 2-1，課題 2-2，課題 3-1，課題 4-1，課題 4-2，課題 4-3，課題 5-1，課題 5-2

一時提出するファイル名

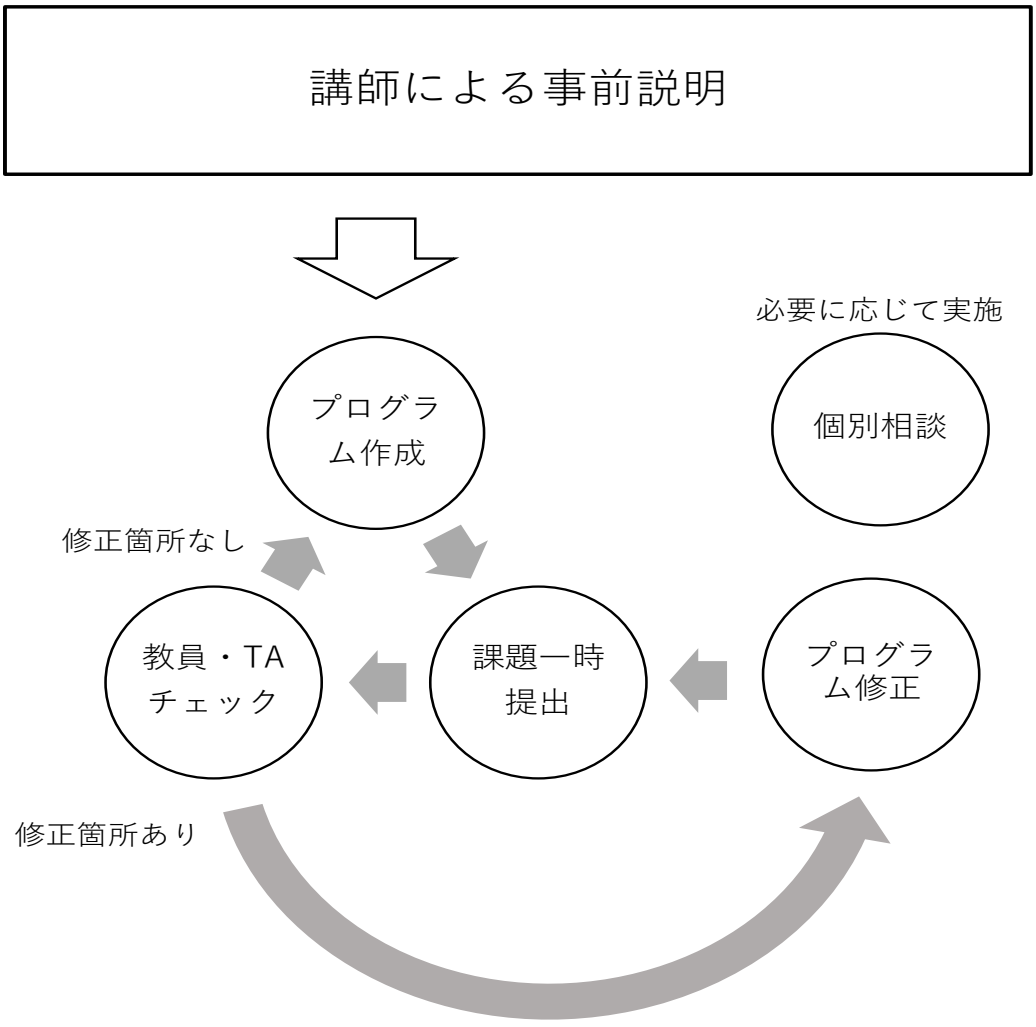
学生番号-氏名-課題プログラム名-提出回数.拡張子

例１：02123456-飯田勝吉-ex110-01.html

例２：02123456-飯田勝吉-ex510-01.zip

注意：一課題につき複数のファイルを提出する場合は、ZIP でファイル群を一つにまとめること。「提出回数」の部分は、01，02，のようにインクリメントしていくこと。

注意：Zoom や slack 上の表示名は「学生番号-氏名」にすること。学生番号・氏名に基づき、当該学生を識別し、一時提出ファイルの確認や議論を行う。（注：学生番号だけだと、人を間違えるリスクが高いので氏名も付与すること）




各回の実験時間の流れ

１．最初に講師による事前説明を実施

２．Zoom につないだまま、個別にプログラム開発（基本的に無音）

３．完成した場合(**)の URL から課題一時提出を実施
４．教員・TA はチェックを実施し、その結果を(***)の URL に記載
５．修正箇所があった場合、個別にプログラムを修正
６．その際、質問があれば Slack から質問
７．それでも解決しない場合、Zoom のブレイクアウトルームを使い個別相談を実施
８．教員・TA チェックに通った場合、次の演習課題に取り組み、３．に戻る

時間中は Zoom につないだままなので､全体説明がある場合に教員から発言することがある。
なお、出席はとらない。合格の条件は
１．すべてのプログラムが課題一時提出で合格すること
２．レポートを提出し合格すること
の２点である。

**実験内容**：

**第１回　JavaScript の基本（１）**

| |
|---|
| 実行環境のセットアップ，まずは“Hello World!”から<br>＜変数，演算子，条件判断，繰返し，配列，関数＞ |

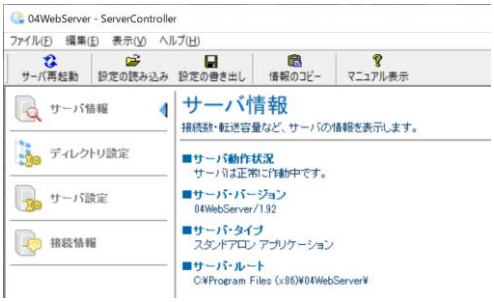
課題 1-0（ローカルウェブサーバ準備）

本実験テーマの全サンプルプログラムを演習ファイル提供 URL (p.2, (*))から各自のコンピュータにダウンロードすること。本演習で扱う JavaScript および HTML のプログラムは「ローカルファイル」としてアクセスする場合と、「ローカルウェブサーバ」を経由して扱う場合がある。

「ローカルウェブサーバ」が必要な課題：課題 4-1, 6-1, 6-2

「ローカルウェブサーバ」の準備方法を以下に説明する（Windows を前提に説明）

１．ローカルウェブサーバプログラムのインストール
http://www.soft3304.net/04WebServer/
から「04WebServer」の最新版(1.92)
（標準インストールでかまわない）




04WebServer の設定画面（特に設定変更は不要）

２．ダウンロードしたファイルの配置

C:¥Program Files (x86)¥04WebServer¥DocumentRoot

に、例えば mwa (モバイルウェブアプリケーションの意)というフォルダを作成し、このフォルダ内にファイルをおく

Mac 利用者向けの情報：

【Mac】ローカルに Web サーバを立ち上げる超簡単な方法！

https://original-game.com/mac-local-web-server/

３．指定ブラウザからローカルウェブサーバにアクセス

http://localhost/mwa/

にアクセスすると、実行可能。

なお、課題 4-1、6-1, 6-2 以外はローカルウェブサーバを必ずしも必要としないので、ブラウザから直接 HTML ファイルを呼び出して実行すればよい。

４．実験で利用する HTML ファイルへのアクセス方法１（ローカルファイル）

演習ファイル提供 URL (p.2, (*))からダウンロードした HTML ファイルをエクスプローラなどでアクセスし、Firefox ESR 上にドラッグアンドドロップすることで実行可能。

５．実験で利用する HTML ファイルへのアクセス方法２（ローカル Web サーバ）

Firefox を起動し、３．で記載した URL にアクセスすると以下のような画面が表示される。

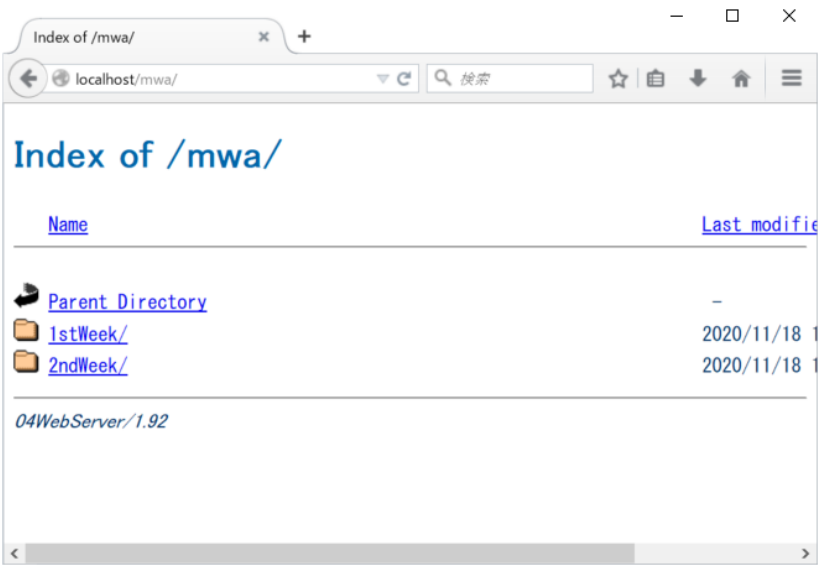



図１：ローカル Web サーバのトップ画面

上記は、C:¥Program Files (x86)¥04WebServer¥DocumentRoot¥mwa の中に配置されているファイル一覧である。 “1stWeek” は、１～３日目の演習プログラムを、 “2ndWeek” ４～６日目の演習プログラムを置いている。一覧の中から HTML ファイルを探してアクセスすることで、４．のローカルファイルと同様にアクセス可能となる。

私物スマホを用いた演習は、私物 PC 内で起動したサーバにアクセスすることが困難であるため、今年度は実施しない。

――実施しない演習：ここから――

６．実験で利用する HTML ファイルへのアクセス方法３（私物スマホ）

実験室の計算機と接続可能な無線 LAN アクセスポイントを配置している。

実験室無線 LAN への接続方法

SSID: iec_wlan_g または iec_wlan_a、パスワード: iec_wlan

私物のスマホ、タブレット等から上記の無線 LAN アクセスポイントに接続する。その後、５．で入力した URL を入力すると、私物スマホ等のブラウザからもアクセスできる。（この無線 LAN は、本実験のみに対して提供される。P2P ファイル交換ソフトの利用など、禁止されている通信は行わないこと）

――実施しない演習：ここまで――

７．Hello World プログラムの実行

実際の課題のプログラムに入る前に、Hello World を表示する HTML/JavaScript のプログラムを表示してみること。サンプルプログラムは ３ つあり、ex100.html, ex101.html,

ex102.html である。ex100.html と ex101.html は HTML ファイルの中に JavaScript のコードが埋め込まれているもので、ex102.html は外部の JavaScript ファイル ex102.js を呼びだす構造となっている。ex100.html は、<body>タグ内に JavaScipt コードのすべてが埋め込まれており、ex101.html は、<head>タグ内に JavaScript の関数が定義されており、それを<body>がタグの中から呼び出す構造になっている。

このように、JavaScript プログラムは HTML ファイルの中に埋め込まれる場合と、別のファイルになっている場合がある。今後の実験課題では二つの場合がそれぞれ含まれているので、確認すること。

また Hello World の動作確認は、４～６．で説明した３つのアクセス方法を試してみること。なお、ex100.html, ex101.html, ex102.html, ex102.js は完成したサンプルコードであるが、これ以降に利用するサンプルコードは完成していないコードである。

８．HTML や JavaScript のソースコードの編集方法

自由なエディタを用いて ex110.html などの配布プログラムの変更作業を行うこと。

課題 1-1

サンプルプログラム ex110.html をもとにして修正・変更を加え，以下の仕様を満たす JavaScript プログラムを完成せよ。

（仕様）

(1) 10 進数（0 以上，255 以下の整数）を 16 進数に変換するプログラムを書け

(2) 上記の条件を満たさない入力に対しては，出力として 0 を返すこと

(3) 与える数値はソースプログラムの中に埋め込み，変換の例を 10 個程度任意に設定し，変換前後の結果をブラウザに見やすく表示すること（例 : 255 → 0xFF）（ここで、0xFF の「0x」は 16 進数を慣例的に表す文字列である）

課題 1-2

サンプルプログラム ex120.html をもとにして修正・変更を加え，以下の仕様を満たす JavaScript プログラムを完成せよ。なお、表の作り方を説明するために 2×2 の表を表示する ex120-sample.html を置いている。これを参考にすること。

（仕様）

(1) 16 進数の掛算表（「九九」の表の 16 進数版）をブラウザに表示すること

(2) 表の行と列はそれぞれ 1, 2, 3, …, 9, A, B, C, D, E, F からなること

(3) 掛算表の列の桁位置をそろえ，できるだけ見やすい形に整えること

(4) 課題 1-1 の変換ルーチンのほか，繰返し，配列，関数を利用して実装すること

(5) 16 進数を表す 0x は省略してよい。
(6) ex120-sample.html は表の枠線等の装飾をつけていない。HTML の表の装飾方法を調査し、さらに見栄えをよくするような工夫をおこなうこと（6 は発展的課題とする。余裕が行うこと）

## 第 2 回　JavaScript の基本（２）

ブラウザのイベント検出と DOM ツリーのアクセス
＜オブジェクト，イベントハンドラ，DOM＞

課題 2-1

サンプルプログラム ex210.html をもとにして修正・変更を加え，以下の仕様を満たす JavaScript プログラムを完成せよ。

（仕様）
(1) ブラウザ上に 2 個の押しボタン A, B を並べて配置すること
(2) ボタン A を押す度に，10 種類の異なる背景色（色の設定は自由）から 1 つをローテーションで順次選択して切り替えること
(3) ボタン B を押すと，ボタン A の場合とは逆方向にローテーションして切り替わること
(4) 初期状態の背景色は白であること
DOM，イベントハンドラの解説を巻末につけるので、参照すること。

課題 2-2

サンプルプログラム ex220.html をもとにして修正・変更を加え，以下の仕様を満たす JavaScript プログラムを完成せよ。この課題はテキスト行を追加・削除する JavaScript プログラムを作るものであるが、参考のためにテキスト行の追加だけを実施するサンプルプログラム ex220-sample.html を提供する。

（仕様）
(1) ブラウザ上に 2 個の押しボタン A, B を並べて配置すること
(2) 初期状態では，ボタン以外に何も表示されていないこと
(3) ボタン A を押す度に，最初の行からの行数（行番号）と何らかのテキスト行（例 : “1 情理太郎”）を，ボタンが配置されていない表示領域に，改行して追加表示すること
(4) ボタン B を押す度に，表示されていた最後の行だけが削除されること
(5) 何も表示されてない状態でボタン B を押しても変化は起こらないこと
(6) 各行で表示するテキストは、行番号以外は同一のものでかまわない。

## 第 3 回　JavaScript の応用

ブラウザで動作する簡単なストップウォッチアプリ
＜Date オブジェクト，イベントハンドラ，DOM＞

課題 3-1

サンプルプログラム ex310.html, ex310.js をもとにして修正・変更を加え，以下の仕様を満たす JavaScript プログラムを完成せよ。

（仕様）

(1) ボタン操作によって時間経過を計測し，時・分・秒単位の数値で表示すること
(2) 時間表示の最小桁は 0.1 秒であること
(3) スタートボタン，ストップボタン，リセットボタンを備えること
(4) スタートボタンで計測を開始し，ストップボタンで計測を停止すること
(5) 計測中は 0.1 秒単位で経過時間表示が更新されること
(6) リセットボタンを押すことで、時間経過の表示を 0 に初期化できること。ただし、計測中はリセットボタンが反応しないようにすること
(7) ストップボタンで計測を一度停止したあと，再びスタートボタンを押すことで，表示をリセットすることなく計測を継続できること（7 は発展的課題とする。余裕がある者が行うこと）

巻末に時刻を扱う関数群 setInterval, clearInterval, Date(), getTime() の解説をつけるので、参照すること。

## 第 4 回　ライブラリ jQuery の基礎

jQuery のセットアップ、セレクタ、省略記法、Ajax、外部 API 利用、アニメーション

課題 4-0（前準備）

３～６日目のフォルダ内にある、ex400.html, ex401.html をブラウザ上で実行し、その動作を確認すること。これらのプログラムは、カーソルが文字の上に載った際に、文字の色が変わり、クリックすると、すぐにページ遷移するのではなく、ポップアップウィンドウがでてきて、遷移してよいかどうか確認される。二つのプログラムの違いは、課題 4-1 で説明する、省略記法を使っているかどうかにある。詳しくは課題 4-1 で説明する。

課題 4-1

jQuery で書かれたサンプルプログラム ex410.html は省略しない記法で書かれている。ex400.html, ex401.html および巻末の解説を参考に省略記法を用いてプログラムを書き直

すこと。その際、できる限り省略して記載すること。

注意：ex410.html は JavaScript プログラムがサーバと通信してファイルを入手するプログラムである。そのため、ローカルファイルとして Web ブラウザに読み込んだ場合、正常に動作しないことがある。実習書前半の 3 ページ「５．実験で利用する HTML ファイルへのアクセス方法２（ローカル Web サーバ)」の方法でアクセスすること。

解説について：jQuery はセレクタと呼ばれる機能が JavaScript に追加されている。また、JavaScript プログラムがファイルを取得する機能に Ajax がある。これらの機能および省略記法については、巻末に解説をつけるので、参考にすること。

課題 4-2（2019 年度更新版）ex420 -> ex421

jQuery で書かれたサンプルプログラム ex421.html は、外部 API を呼び出すプラグインソフトウェア(memewaza_weather)を利用したサンプルプログラムであり、省略記法で書かれている。ex400.html, ex401.html および巻末の解説を参考に省略を用いない書き方でプログラムを書き直すこと。その際、すべての省略記法を省略しない書き方に書き直すこと。また、以下の仕様を満たすような改造を施すこと。

（仕様）

(1) 公式 Web サイトをみて、天候を表示する都市を変更すること。
(2) 公式 Web サイトをみて、天気予報の期間及び表示形式を変更すること。
(3)公式 Web サイトをみて、天気予報の詳細説明（天気概況）を表示するよう変更すること。
　　(3)は廃止したため実施は不要。
(4) (発展課題) ボタンを押すことで、天候情報を表示する都市を変更できるようにすること。

mamewaza_weather プラグインソフトウェアの公式 Web
https://mamewaza.com/tools/weather.html

解説について：外部 API からリアルタイムに情報を収集することについて、巻末に解説をつけるので、参考にすること。

課題 4-3

jQuery で書かれたサンプルプログラム ex430.html は、スタートボタンを押すとイラストがパラパラ漫画のようなアニメーションとして動き出し、次にボタンを押す（ボタン内の表示はストップに変わっている）と、アニメーションが停止する。

サンプルプログラム ex430.html に修正・変更を加え，以下の仕様を満たす JavaScript プ

ログラムを完成せよ。

（仕様）

(1) アニメーション停止状態で、ウェブ画面上のいずれかの点をダブルクリックすると、静止しているイラストがダブルクリックした点をめがけて滑らかに動き出し、ダブルクリックした点に到達すると静止させること。

(2) アニメーション実行状態で、ウェブ画面上のいずれかの点をダブルクリックすると、アニメーションを一時停止させ、その状態で(1)と同様にダブルクリックした点までイラストを移動させ、移動終了したのちに、アニメーションを再開させること。

(3) 上記二つの仕様を満たすために、サンプルプログラム ex430.html 内の「Web ブラウザ上をダブルクリックしたときに呼ばれる関数」の中身を記載すること。

解説について：巻末に jQuery でアニメーションを行う方法、およびクリックやダブルクリックした点の座標情報を取得する方法について記載するので、参考にすること。

## 第 5 回　ライブラリ jQuery Mobile の基礎

| |
|---|
| jQuery Mobile のセットアップ、フォーム、スワイプ |

課題 5-1

jQuery Mobile で書かれたサンプルプログラム ex510.html は、jQuery Mobile のフォーム入力の基本要素をまとめたものである。これに基づき、以下の作業を行うこと。

(作業)

(1) いずれかの企業、大学、組織等の Web サイトを想定し、そのサイト内のフォーム入力が必要な Web ページを設計すること。レポートには、どのような Web サイトを想定したのか、そしてどのようなフォーム入力が必要なのか、そのページの仕様を記載すること。

(2) (1)の仕様を満たす Web ページを構築すること。その際、ex510.html に含まれているフォーム入力の基本要素のうち、3 つ以上の要素を利用すること。

(3) (発展課題) PC ブラウザとスマートフォン、タブレット等で、操作性がどのようにかわるかを考察すること。スマートフォンの操作に関しては、シミュレータを利用して確認すること。レポートには、シミュレータの画面を添付すること。

(4) (発展課題) jQuery Mobile のテーマローラサイトを用いて、自作したテーマに変更すること。レポートには、自作したテーマに基づくプログラムの実行画面を添付すること。

ThemeRoller For jQuery Mobile

http://themeroller.jquerymobile.com/

シミュレータの利用方法：

「Google Chrome ブラウザでスマートフォンサイトをチェックする」

http://www.atmarkit.co.jp/ait/articles/1403/20/news050.html
（注：スマートフォンのシミュレータ機能は Firefox ESR にもあるが、機能が限定的なので、この演習に限り Google Chrome の利用を推奨する）

課題 5-2

モバイルデバイスと PC の操作性の違いにスワイプがある。ex520.html はモバイルデバイス向けにスワイプを実現するサンプルコードで、ex521.html は PC 向けにマウスクリックで左右の情報にアクセスさせるサンプルコードである。以下の仕様を満たすプログラムを作成すること。

(仕様)

(1) PC およびモバイルデバイス双方に対応できるよう、スワイプおよびマウスクリックの双方で操作するプログラムを構築すること。その際、ex520.html と ex521.html をマージすること。

(2) PC とモバイルデバイス双方に対応する Web サイトを構築する際に、気を付けるべきことを考察すること。

## 第 6 回　JavaScript のセキュリティ（DOM based XSS）

| |
|---|
| DOM based XSS 脆弱性、情報の搾取、URL フラグメント、<br>情報の書き換え、ソースとシンク、エスケープ、サニタイジング、<br>安全なシンク関数、バージョン管理 |

課題 6-1

ex610.html は、URL フラグメントを付与して Web ブラウザを呼び出すと、その内容が Web 画面上にでてくる。

正常の利用例：http://192.168.X.X:8080/2ndWeek/ex610.html#lida


図 3：ex610.html の実行画面（URL はテキストのみ）

ここで、URL フラグメント部分に JavaScript のコードを埋め込むと、実行されてしまう。
（注意：URL が http://192.168.X.X:8080/・・・となっているが、今年度の演習では、

http://localhost/mwa/・・・となるため、適宜置き換えること。

不正コードを埋め込んだ例：
http://192.168.x.x:8080/2ndWeek/ex610.html#"><img src=x onerror=alert('Danger!')></img>

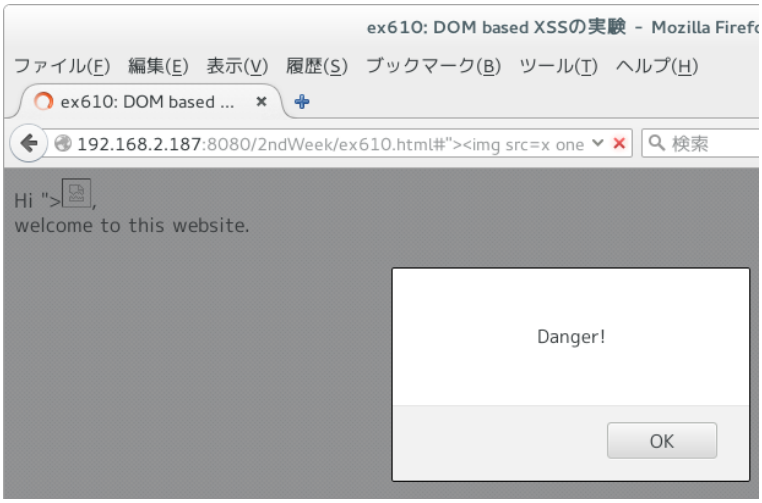



図 4：ex610.html の実行画面（URL に JavaScript コードを埋め込んだ例）

巻末の解説に記載したとおり、この脆弱性は DOM based XSS 脆弱性と呼ばれる。ex610.html の DOM based XSS 脆弱性に関して、以下の調査を行うこと。

(調査の内容)
(1) 演習のために特別にインストールした Firefox 38.8.0 ESR だけでなく、Firefox, Chrome など、通常利用しているブラウザで不正コードを埋め込んで ex610.html を実行することで、ブラウザ上で JavaScript が実行されるかどうかを調査せよ。レポートには、調査結果を示す画像を添付すること。（Firefox 38.8.0 ESR 以外のブラウザは、最低限一つ試すこと）
(2) (発展課題) スマホ、タブレット上で(1)と同様の調査を行うこと。レポートには、調査結果を示す画像を添付すること。（PC で立ち上げたローカルウェブサーバにスマホからアクセスできない場合は、この課題を実行しなくてよい）

課題 6-2

ex620.html は、成績管理 Web サイトをモデルに作った Web ページである。成績を管理する教科を変更可能とするために、ex610.html と同様に URL フラグメントを用いている。

正常の利用例：http://192.168.X.X:8080/2ndWeek/ex620.html#プログラミング 1


図 5：ex620.html の実行画面（URL はテキストのみ）

ここで、課題 6-1 と同様に URL フラグメントに JavaScript コードを含むことができる。JavaScript の機能を使えば、外部の第 3 者がウェブ画面に書かれている成績情報を窃取することが可能となる。

（＊1）不正コードを埋め込んだ例：

http://192.168.x.x:8080/2ndWeek/ex620.html#"><img src=x onerror=******></img>

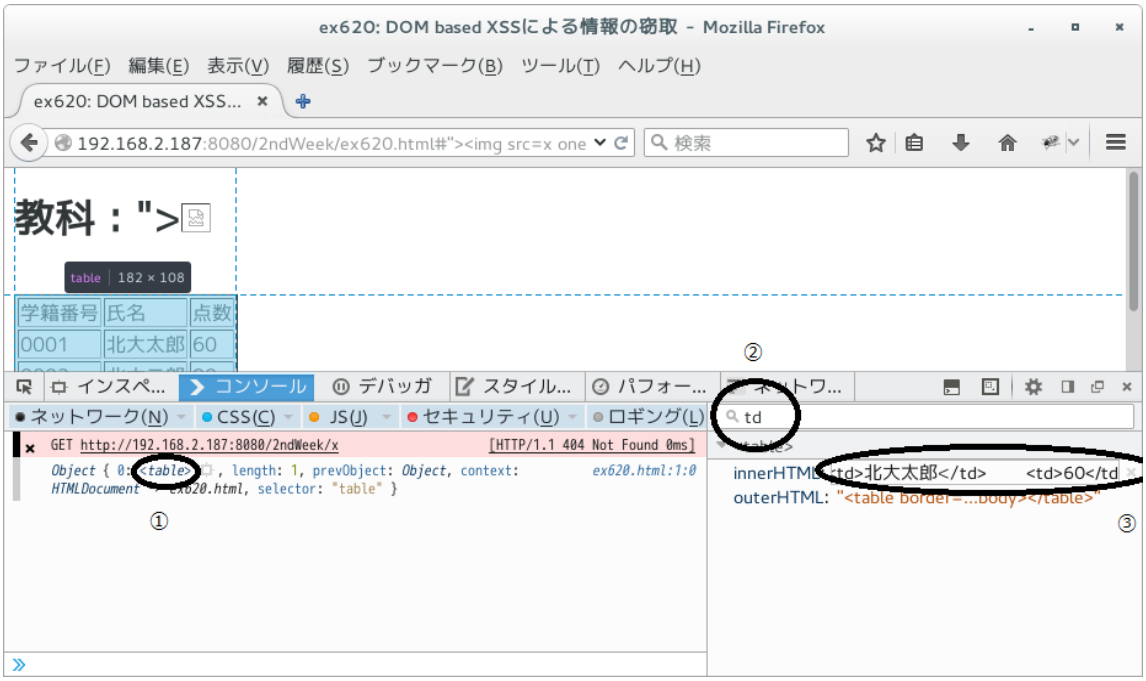



図 6：ex620.html の実行画面（URL に JavaScript コードを埋め込んで情報窃取をする例）

ここで、本当の攻撃者は何らかの外部の場所に窃取した情報を送信するはずである。ここで

は、同じブラウザ上のコンソール画面を外部の送信先と仮定して、（＊１）の******の部分を追記して以下の URL を用いて攻撃するものとする。

（＊２）不正コードを埋め込んだ例：
http://192.168.x.x:8080/2ndWeek/ex620.html#"><img src=x onerror=console.log(#)></img>

注意１：console.log()は Web ブラウザのコンソール画面に文字列を出力する関数である。Firefox の開発ツールのコンソール画面を選択することで、出力された文字列を確認できる。

(仕様)
(1)（＊２）の console.log(#)の#部分を書き直し、成績情報をコンソール画面に出力する URL を導き出せ。その際、通常の JavaScript の機能を用いること。
(2)（＊２）の console.log(#)の#の部分を書き直し、成績情報をコンソール画面に出力する URL を導き出せ。その際、jQuery の機能を用いること。

注意２：この実験課題を用意した理由は、JavaScript の機能の悪用方法を教授するためではない。安全なコードを書くためには、どこが脆弱で、どのようにすれば安全になるかを学習するためである。レポートには、この実験課題で学習したことを悪用せず、不正アクセス禁止法などの日本の法律に違反する行動を行わないことに関する宣誓の文言を記載する事。
注意３：レポートには、情報を窃取できたことを示す画面を添付すること。

（課題 6-3：今年度は実施しない）
過去のサンプルプログラム ex420.html は脆弱性を含んだコードである。本資料 30～32 ページに記載したセキュアプログラミングのガイドを参考に、安全なコードに書き直すこと。また ex620.html を対象に、脆弱性を含まないように書き直した ex630.html も提供するので、これも参考にすること。

注意：ex410.html は外部の HTML ファイルを読み込んでいるので、脆弱性を含む可能性がある。しかし、中で読み込んでいる ex410-load.html に不正なコードが入り込まないように管理できる範囲においては、必ずしも改善が必要とはいえない。もし、ex410-load.html が第３者によって書き換え可能な場合は、それを DOM based XSS の「ソース」と認識することが必要で、そのうえで ex410.html の改善が必要となる。

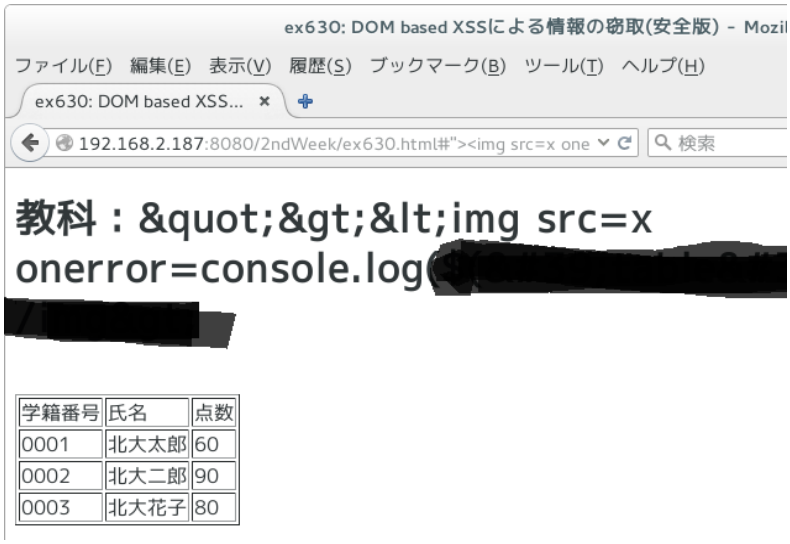



図 7: ex630.html の実行画面（不正コードが実行されていない）

課題 6-4（発展課題）

DOM based XSS による被害を軽減するためにはどのような活動を行うことが有効かを考察せよ。具体的には、巻末の解説に記載した安全なコードの原則を自分なりにまとめ、そのうえで、脆弱なコードを減らし、安全なコードを増やしていくために

・プログラマに対する教育活動

・技術的に安全にする方法の研究開発

等の活動を行うことが必要となる。これらのことを考察し、レポートに記載すること。（参考文献(7～11)が参考になる。中でも参考文献(10)が参考になる）

---

**用語集**：

○ JavaScript

JavaScript は, クライアントサイドスクリプト言語で, Web ブラウザが読み込んだ HTML ファイル内のコードを，Web ブラウザ自身が解釈・実行します。これによって以下のようなインタラクティブな機能を持った Web コンテンツを作成することができます。

(1) HTML 要素やその属性値を動的に変化させる

(2) ブラウザ上のマウス操作に反応してメッセージを表示する

(3) Web サーバからデータを取得して，ブラウザ上の情報を動的に変化させる

JavaScript は Java と名前が似ていますがまったく別物です。JavaScript を省略して Java と呼ぶのは絶対にやめましょう。なお，サーバサイドスクリプト言語は Web サーバ上で実行されるもので，PHP，Ruby，Perl などがあります。

○ Ajax

Ajax（Asynchronous JavaScript and XML）（エイジャックス）は，ブラウザと Web サーバがバックグラウンドでデータをやりとりする非同期通信の仕組みです。ページを更新あるは再読込することなく Web ページのコンテンツを書き換えたり，サーバからその都度取得したデータをポップアップでページ上に可視化したりできます。Google マップは Ajax の最も身近な事例です。ブラウザ上で地図をドラッグすることで，足りない領域の地図データだけが非同期に Web サーバから読み込まれ，ブラウザに表示されます。

○ jQuery

jQuery（ジェイクエリー）は JavaScript ライブラリで，以下の特徴を持っています。

(1)“Write less, do more”（少ないコードで沢山のことを）

(2) 基本的なページ操作から，イベント処理，エフェクト，非同期通信，標準 JavaScript の拡張など，JavaScript による Web ページ開発に必要な機能をサポート

(3) Chrome，Firefox，Opera，Safari 5.1 以上，IE 6 以上，iOS 6.1 以上，Android 2.3/4.0 以上など，主要ブラウザを幅広くサポート

(4) jQuery UI や jQuery Mobile などのプラグインが豊富

利用統計によれば,インターネット全体の 35%ものサイトで jQuery が利用されており，jQuery が JavaScript ライブラリのデファクトスタンダードとなっています。また jQuery では，必要なライブラリを CDN（Content Delivery Network）でも提供しています。

○ jQuery Mobile

jQuery Mobile は，jQuery を拡張するライブラリ（プラグイン）の一種で，スマートフォンやタブレットに代表されるモバイル対応アプリケーションを開発するための以下のような機能を提供します。

(1) アプリケーションのデザインを管理するテーマ機能とリッチなフォーム要素

(2) ダイアログ，リスト，パネル，レスポンシブルテーブルなどの各種ウィジェット

(3) Ajax 通信（Web サーバとの非同期通信）によるページ遷移

(4) デバイスのタッチパネル環境に対応した様々なイベント処理

これらの機能を利用することで，アプリケーション開発者はデバイスに依存することなく，手軽にネイティブアプリライクな UI を実装することができます。

JavaScript のリファレンス：

本実験では、HTML と JavaScript をつかった Web プログラミングを実施し、その中で重要なのは JavaScript である。そこで本節では、JavaScript の中の重要な関数等を解説する。

○ DOM (Document Object Model)

JavaScript を使うことで Web ページの内容の追記・修正・削除等ができる。そのための基盤として利用するのが DOM である。DOM は、単一のウェブページ(=HTML ドキュメント)を木構造で表現するモデルである。その木構造の一部の要素を選択し、選択した要素に対して修正･追加等を行う関数が JavaScript で提供されている。以下､DOM の例と DOM を操作する関数群を紹介する。

図 9 にサンプル HTML を示し、サンプル HTML に対応する DOM ツリーを図 10 に示す。このように DOM では、HTML ドキュメントを木構造であらわす。ここで、木構造の各要素を「エレメントノード」あるいは単に「ノード」と呼ぶ。頂点は、<html>エレメントノードの上に配置された document エレメントノードである。

```
<html>
  <head>
    <title>ex210: 背景色を書き換える</title>
    <style>中略</style>
    <script>中略</script>
  </head>
  <body>
    <div>
      <button onclick="changeBgForward()">背景色を変更する（前方）</button>
      <button onclick="changeBgBackward()">背景色を変更する（後方）</button>
    </div>
  </body>
</html>
```

図 9：サンプル HTML

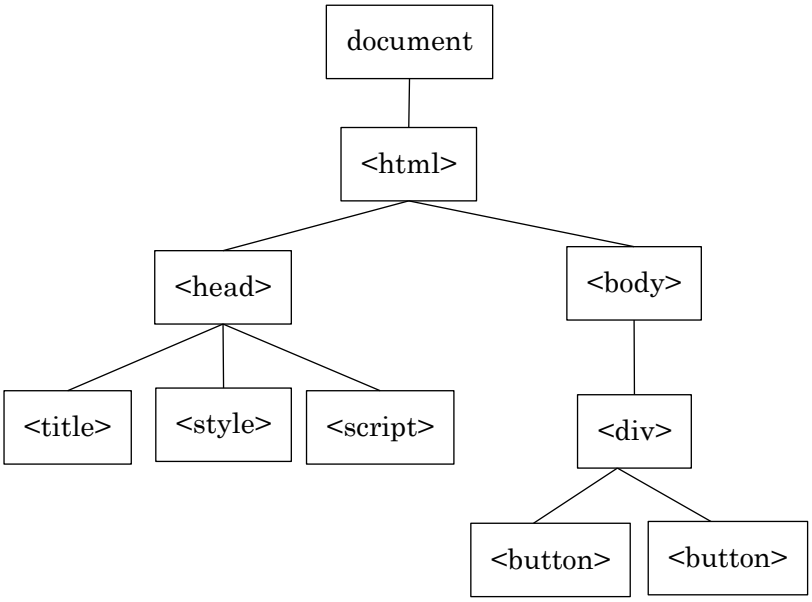



図 10：サンプル HTML の DOM ツリー

JavaScript の DOM を扱う関数群は、DOM ツリーからノードを探索する関数群（表 1）と実際にノードを作成・追加・削除を行う関数（表 2）に分けられる。各関数の記述方法は ex220-sample.html を参考にすること。

また、ex210.html で document.body.style.backgroundColor に色コードを代入しているが、これは、DOM ノードの内容を修正する例である。

表 1：DOM ツリーからノードを探索する関数群

| 関数名 | 機能 | 利用できるノード |
|---|---|---|
| getElementById(id) | ノードの識別子が Id であるノードを探索 | document のみ |
| getElementsByTagName(tag) | タグ名が tag であるノードを探索 | 任意のエレメントノード |
| getElemetsByClassName(className) | クラス属性値が className であるノードを探索 | 任意のエレメントノード |
| getElementsByName(name) | 名称に name を含むノードを探索 | document のみ |
| querySelector(selector) | セレクタ名が selector であるノードを探索 | 任意のエレメントノード |
| querySelectorAll(selector) | セレクタ名が selector | 任意のエレメントノ |

| | | |
|---|---|---|
| | であるノードをすべて探索 | ード |

表 2: DOM ツリーのノードを操作する関数群

| 関数名 | 機能 | 利用できるノード |
|---|---|---|
| createElement(tag) | タグ名が tag であるノードの作成 | document のみ |
| appendChild(element) | createElement()で作成したノードを自身のノードの子として加える | 任意のエレメントノード |
| removeChild(element) | 自身の子ノードである element を DOM ツリーから削除する | 任意のエレメントノード |

○ イベントハンドラ：

マウスのクリック等のイベントの発生をプログラムに通知する仕組みをイベントハンドラと呼ぶ。課題 2-1、2-2、3-1 では、ボタンが押されたというイベントを JavaScript に通知するためにイベントハンドラを使っている。

――イベントハンドラの利用方法：ここから――

１．HTML 側から関数を指定する方法（課題 2-1）：

<button onclick=”changeBgForward()”>

２．JavaScript 側からイベントを指定する方法（課題 2-2, 3-1）：

addButton.addEventListener(“click”, addText, false)

――イベントハンドラの利用方法：ここまで――

addEventListener()関数は、引数として、イベント名、イベントを検出した後に呼び出す関数名、キャプチャリングフラグの３つを指定する。（キャプチャリングフラグは false を渡せばよい。）上記の２．の例は、addButton という DOM ノードにおいて”click”イベントが発生した場合、addText という関数を呼び出せという命令を意味する。なお、イベントハンドラを解除する方法として、removeEventListener()関数が存在する。

○時刻を扱う関数群：

課題 3-1 では、ストップウォッチを作成するために、時刻を扱う関数が必要になる。ここでは、それらの関数群を説明する。

setInterval()

タイマーID = setInterval(定期的に実行したい関数名，実行する間隔[ms])

説明：第２引数で指定する間隔で、第１引数で指定する関数を定期的に実行する。

clearInterval()

clearInterval(タイマーID)

説明：引数で指定したタイマーID のタイマーを解除する

Date()

Date オブジェクト= new Date()

説明：実行時の時刻（1970 年 1 月 1 日午前 0 時(UTC)からの経過時間）を示す Date オブジェクトを生成する。

getTime()

Date オブジェクト.getTime()

説明：1970 年 1 月 1 日午前 0 時(UTC)からの経過時間をミリ秒単位の整数値で示す。

**解説**：

本実験では、jQuery、jQuery Mobile そして、DOM based XSS について取り扱う。これらの課題を行うために必要または有用となる以下の事項を解説する。

・JavaScript のデバッグ：Firebug

・jQuery の解説（セットアップ、セレクタとメソッド、省略記法、Ajax の記述方法、外部 API の利用、アニメーション、クリックイベントの座標）

・jQuery Mobile の解説（セットアップ）

・DOM based XSS の解説（XSS 脆弱性とその脅威、URL フラグメント、DOM based XSS の基礎、JavaScript のセキュアプログラミング）

○JavaScript のデバッグ：Firebug

Firefox を使って JavaScript をデバッグするには、Firefox に内蔵されている開発ツールが有用である。より高度なデバッグを可能とする追加モジュールに Firebug がある。もし、なかなかバグがとれない場合は、Firebug を導入して確認すること。

○jQuery の解説：セットアップ

jQuery はライブラリであるので、利用するためには HTML ファイル内で参照する必要がある。参照の方法としては、ライブラリをダウンロードして各 Web サーバに配置する方法があるが、CDN と呼ばれる配布方法を用いて参照する方式が普及している。具体的には

（＊３）jQuery.com の CDN から読み込む場合

```
<script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>
```

（＊４）Google の CDN から読み込む場合

```
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.0/jquery.min.js"></script>
```

というコードを HTML の<head>もしくは<body>内に記載すればよい。なお、現在 jQuery は 1 系、2 系、3 系の 3 バージョンが並行して提供されている。1 系と 2 系の（セキュリティパッチ提供以外の）開発は終了しており、今後は 3 系への移行が促されている。そのため本演習では、2019 年から最新版の 3 系を利用することとした。（2020 年 9 月 9 日現在、最新版は 3.6.0）。DOM based XSS のセキュアプログラミングの解説で述べるとおり、最新版を利用しないことによるセキュリティ被害がでた事例がある。1 系の開発途上において、常に最新のバージョンを提供する “jquery-latest.js” という名称のファイルが配布されていたことがあった。「latest」版を使うことで、セキュリティ被害を最小化することができるが、しかし、バージョンアップの際のサービス提供ができなくなる別の被害が生じたため、「latest」という名称での最新版の提供は行われなくなった。そのため、開発者は定期的にバージョンアップ状況の確認と、バージョンアップが行われた場合は自分のコードが新しいバージョンの jQuery ライブラリでも動作可能かの確認などを行う必要がある。

なお、ファイル名の min は、人間にとっての可読性を高めるためのスペース、改行等を取り除いた最小コードを指定するものである。HTML から jQuery を指定する場合は min を指定すればよい。

○jQuery の解説：セレクタとメソッド

JavaScript の機能の一つに、DOM ツリーからノードを探索し、見つかったノードに対して操作を行なうことがある。その際、ノードを探索する関数が多数あって、どの関数を使えばよいか、毎回適切な選択が必要であった。そのため jQuery では、汎用の探索関数である jQuery ()を提供する。一般的な jQuery の関数は、以下の表記方法で提供される

jQuery(セレクタ).jQuery のメソッド名()

上記の jQuery(セレクタ)の部分で DOM ノードを選択し、選択された DOM ノードに対し、何らかの処理を行うのが jQuery のメソッドである。以下では、主に jQuery のセレクタについて説明する。

```
<body>
  <div id="header">
    <h1>jQueryについて</h1>
    <p id="description"><strong>jQuery</strong>は便利なJavaScriptライブラリ。</p>
  </div>
  <div id="contents">
    <h2>jQueryの特徴</h2>
    <ul id="list">
      <li class="myClass">クロスブラウザに対応</li>
      <li class="myClass">豊富な<strong>エフェクト</strong>を提供</li>
      <li><strong>アニメーション</strong>も簡単に実現</li>
      <li><span id="mySpan">デザイナー</span>に人気の<em>jQuery</em></li>
      <li>デザイナーに人気の<em>jQuery</em></li>
    </ul>
  <div>
</body>
```

図 11:サンプル HTML コード

図 11 のサンプル HTML コードに対する jQuery セレクタの例を以下に示す。

１．jQuery(“h1”)　　＝　<h1>タグを選択する要素セレクタ
２．jQuery(“#mySpan”)　＝　“mySpan”という ID を選択する ID セレクタ
３．jQuery(“.myclass”)　＝　“myClass”というクラスを選択するクラスセレクタ
４．jQuery(“div#contents”)＝　タグが で、ID が contents の要素を選択
５．jQuery(“li.myclass”)　＝　タグが<li>で、クラスが myClass の要素を選択
６．jQuery(“li strong”)　＝　タグが<li>で、その内側にあるタグの要素を選択
７．jQuery(document)　＝　HTML document 全体を選択

他にも、セレクタの指定方法があるので、詳細は jQuery の参考文献、リファレンスマニュアル等を参照すること。

jQuery のメソッドはたくさんあるが、一番利用されるメソッドである

jQuery(document).ready(外部関数);

ready メソッドを説明する。Web ページは、HTML と JavaScript から構成されるが、HTML のレンダリング構築が終わる前に、JavaScript の命令が実行されると困る場合がある。その際に HTML のレンダリング構築が終わったのちに、JavaScript の命令を実行させるのが、ready メソッドである。ready メソッドは引数をひとつとり、外部関数を指定する。つ

まり、HTML のレンダリング構築が終わったのちに外部関数を実行させるのが、ready メソッドである。

○jQuery の解説：省略記法

jQuery では、セレクタとして jQuery()という関数が多数でてくるので、コード行数が長くなる。そこで、多数の省略記法が開発されている。現在利用されているほとんどの jQuery のコードは省略記法が用いられている。そのため、省略記法について説明する。

$省略記法

１番目の省略記法は、記号「＄」を使って省略する記法である。

| 省略しない記法 | 省略記法 |
|---|---|
| jQuery("#mySpan") | $("#mySpan") |
| jQuery(document) | $ |
| jQuery.sample() | $.sample() |
| jQuery(document).ready() | $() |

このように、$省略記法は文脈によって４種類の省略方法がある。また、$.ready()のように、省略記法を部分的に使うことも可能である。

無名関数

他の省略記法に「無名関数」がある。無名関数とは、関数の引数に関数の名称を与える場合に利用できる省略記法である。ready メソッドでは、HTML のレンダリング構築が終わったのちに引数で与える外部関数を実行させるが、無名関数では、外部関数を定義せずに、その場で定義する。具体的には、外部関数の場所に function(){関数の内容}と関数の定義そのものを引数として与える。この説明だけではわかりにくいので、以下に例を示す。

省略しない記法

```
jQuery(document).ready(init);
function init () {
  jQuery("div#contents")..css("background", "yellow");
}
```

省略記法

```
$(function() {
  $("div#contents").css("background", "yellow");
});
```

実際の省略の仕方については、ex400.html と ex401.html を参考にすること。

○jQuery の解説：Ajax の記述方法
Ajax は、JavaScript のプログラムが外部コンテンツを取得するもので、Google Map などで利用されている。ここでは、jQuery の Ajax の記述方法を説明する。

jQuery.ajax(‘http://abc.com/sample.html’).done(外部関数１).fail(外部関数２);

上記は jQuery のバージョン 1.5 以上で有効な記法である。jQuery.ajax(引数)で、引数として URL の文字列を指定する。ダウンロードに成功した場合は、done()で指定する外部関数１が実行され、失敗した場合は fail()で指定する外部関数２が実行される。

ここで、done メソッドと fail メソッドがピリオドでつなげて記述されている。これをメソッドチェーンと言い、計算速度を速めるためにも有効であり、Ajax 以外でも利用可能である。

また、jQuery.ajax() は $.ajax() と省略することが可能である。

○jQuery の解説：外部 API の利用
JavaScript/jQuery の利点は、リアルタイムに更新される情報を取得できることである。たとえば、Google Map では、地図の表示範囲を動かすと、それに追随して地図が動く。また、リアルタイムに情報が更新されるものとして、
・為替レート、株価
・ヘッドラインニュース
・Twitter, Facebook 等のタイムライン
・天候情報、災害等の緊急情報
などがある。これらの情報を、JavaScript から取得し、様々な用途で利用することが可能である。その中で、今回の課題では天候の情報を気象庁の API から取得する mamewaza_weateher を利用した。

しかし、それらの API には、使用条件等が決まっているため、それに沿った使い方をする必要がある。さらに、提供者の都合で利用方法が変わったり、急に提供されなくなったりすることがあるので、注意が必要である。

実際、2018 年度までは Yahoo! Weather API が提供するデータを使う SimpleWeather.js を使っていたが、Yahoo! Weather API の利用条件が変更された（変更日：2019 年 1 月 3 日）ことに伴い、2019 年度から Livedoor Weather Hacks というところから情報を入手する

mamewaza_weahter にプログラムを利用した演習に変更した。さらに、Livedoor Weather Hacks が 2020 年 7 月末にサービス提供が終了したことに伴い、mamewaza_weahter は気象庁のデータを利用するソフトウェアに変更され、今年度は気象庁のデータを利用することとした。

気象庁による気象データ API の利用上の留意事項
https://www.data.jma.go.jp/developer/ryuui.pdf

○jQuery の解説：アニメーション
課題 4-3 では、アニメーションを取り扱っている。本節では、アニメーションを説明する。

１．画像の入れ替えの方法
<img id="myImg" src="images/pic1.png"></img>

という HTML タグの画像を JavaScript から入れ替えるためには、

$("#myImg").attr("src", "images/pic2.png");

を実行すればよい。これを JavaScript のタイマー関数を使って周期的に実行すれば、パラパラマンガが実現できる。

２．画像の移動方法：animate()メソッド
animate()メソッドは、画像などの DOM 要素を変化させるものである。

$(セレクタ).animate(CSS プロパティ, 時間, 変化のスピード);

第 1 引数の CSS プロパティは、JSON 形式という形式で指定する。例えば
{"left": "+=100px", "opacity": "0"}
と指定すると、左に 100 ピクセル移動し、透明度を 0 に変化させることを意味する。
第 2 引数は、ミリ秒単位の数値、または、"slow", "normal", "fast"のいずれかを指定する。
第 3 引数は、"linear"または"swing"が指定できる。前者は一定速度を意味し、後者は最初はゆっくりで次第に速くなることを意味する。

注意：移動のアニメーションを行うためには、CSS の position プロパティをデフォルトの「static」以外の値に変更する必要がある。position: relative （相対位置）または position: absolute （絶対位置）を指定すればよい。

CSS プロパティの設定例

```
#myImg {
   position: relative;
}
```

animate()メソッドの利用例：

```
$(“#myImg”).animate({“left”: “500px”, “opacity”: 0}, “slow”, “swing”);
$(“#myImg”).animate({“left”: “+=100px”}, “slow”, “linear”);
```

○jQuery の解説：クリックイベントの座標

課題 4-3 では、ブラウザ上のクリックした場所の座標を取得する必要がある。

$(セレクタ).click(外部関数１);

$(セレクタ).dblclick(外部関数２)；

jQuery では、クリックイベントまたは、ダブルクリックイベントを上記の関数で取得できる。外部関数は引数としてイベント情報をとる。たとえば、

$(document).click(function(event) {関数の中身});

と書くと、event の中に、クリックした場所の情報が記録される。

event.pageX = X 軸の座標値

event.pageY = Y 軸の座標値

○jQuery Mobile の解説：セットアップ

jQuery Mobile は、jQuery の拡張ライブラリなので、jQuery と jQuery Mobile の二つを参照する必要がある。さらに、専用のスタイルシートの導入も必要となる。

――ここから――

```
<!-- 2021 年 9 月 9 日現在の jQuery Mobile の最新安定板は 1.4.5 -->
<link rel="stylesheet"
href="https://ajax.googleapis.com/ajax/libs/jquerymobile/1.4.5/jquery.mobile.min.css" />

<!-- jQuery 最新版を導入-->
<script type="text/javascript" src="https://code.jquery.com/jquery-3.6.0.min.js"></script>

<!-- 2021 年 9 月 9 日現在の jQuery Mobile の最新安定板は 1.4.5 -->
<script   src="https://ajax.googleapis.com/ajax/libs/jquerymobile/1.4.5/jquery.mobile.min.js">
</script>
```

――ここまで――

○DOM based XSS の解説：クロスサイトスクリプティング(XSS)脆弱性とその脅威

JavaScript はコードの書き方が悪いと DOM based XSS という脆弱性が生じてしまう。DOM based XSS を説明する前に、一般の XSS 脆弱性を説明する。

XSS 脆弱性とは、Cross Site Scripting 脆弱性の略語で、いずれも JavaScript を利用した攻撃を可能とする脆弱性である。すなわち、JavaScript はユーザのブラウザ上で実行されるプログラムであるが、

「ユーザや Web サイトの開発者、JavaScript の開発者が意図していない、不正なコードを実行させること」

ができる場合に、XSS 脆弱性が存在するという。XSS 脆弱性には、Web サーバ側でのプログラムに脆弱性があることによって生じるものと、ブラウザ側のプログラム（つまり JavaScript）に脆弱性があることによって生じるものがある。前者は、たとえば、ある Web ブラウザの利用者が攻撃対象者となってしまい、なんらかの方法で以下の URL(**)にアクセスさせる。

URL(**)

http://XSS 脆弱性のあるサイト A/?q=<script>alert(‘danger!’);</script>

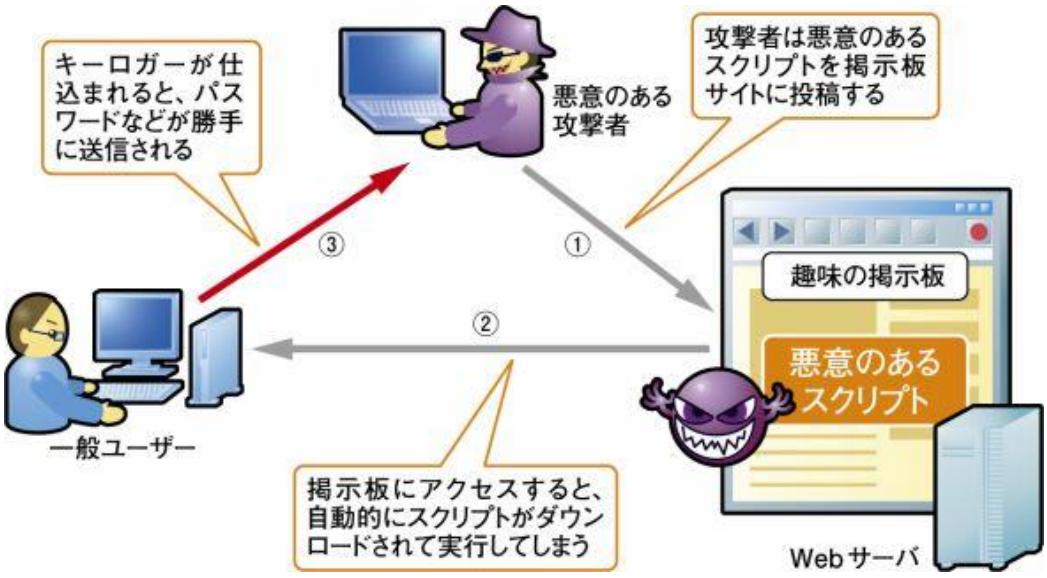



図 12: Web サーバ側の脆弱性を悪用した XSS の例

(http://ascii.jp/elem/000/000/423/423682/index-2.html より引用)

URL(**)の「?」以降の文字列は、URL パラメータやクエリ文字列と呼ばれ、Web サーバ側

のプログラムに対する入力になる。もし、Web サイト A に脆弱性がなければ、URL パラメータはただの文字列として処理されるが、そうでなければ、最終的に攻撃対象者のブラウザ上で実行されてしまう。そのような Web サイトを XSS 脆弱性のある Web サイトと呼ぶ。

一方、DOM based XSS は、正当な目的で提供されている JavaScript に脆弱性があり、それを悪用して不正なコードの実行を可能とする脆弱性である。これについては、のちの解説で説明する。

次に、XSS 脆弱性の脅威について説明する。いずれの脆弱性も、利用者のブラウザ上で不正な JavaScript コードを実行させることを意図している。その結果として
１）攻撃対象者のブラウザ等に不正コードをインストールさせる
２）攻撃対象者のブラウザを介して機密情報を窃取する
３）攻撃対象者のブラウザで表示されている情報を不正に書き換える
ことが可能となる。たとえば、１の攻撃が成功すると、被害を受けた PC が真の攻撃者のために他者を攻撃するための踏み台として利用されたり、キーロガーが仕込まれたりする。２の攻撃が成功すると、攻撃者による不正ログインを許したり、クレジットカード番号、パスワード等の重要情報が漏えいしたりする可能性がある。３の攻撃が成功すると、オンラインバンキング等の Web 上に表示されている重要情報の不正な書き換えが可能になる。

課題 6-2 は、２の攻撃による危険性を体験するための演習である。課題 6-2 では、Web 画面上に記載されている重要情報を対象としているが、Web ブラウザが記憶していてユーザが直接意識することがないクッキーという情報を窃取することも可能となる。クッキーには、何らかの Web サービスの認証が成功したことを示す秘密のランダム文字列を埋め込むことがあり、それが漏えいした場合、攻撃者が成りすましにより Web サービスにアクセスする可能性がある。

○DOM based XSS の解説：URL フラグメント
　Web のアドレスである URL には、特殊な文字列が含まれることがある。URL(**)には、サーバが解釈する特別な文字列を示す URL パラメータの開始を示す記号として「?」が含まれている。一方、クライアント側、つまり、JavaScript に何らかの情報を伝えるものが URL フラグメントである。

URL フラグメントを含む URL の例（431289 が URL フラグメント）
http://example.co.jp/sample.html#431289

上記の例のように、URL フラグメントの開始記号として「#」が使われる。#以降の文字列

は、Web サーバ側は何も処理をせず、ブラウザ上の JavaScript が処理をする。(#から始まることから、URL のハッシュと呼ばれることもある)

URL フラグメントに JavaScript のコードが埋め込まれた場合、DOM based XSS が生じることがあるので、注意が必要である。

○DOM based XSS の解説：DOM based XSS の基礎

先に説明した通り、DOM based XSS は、クライアント側のプログラムである JavaScript のコードに原因となる脆弱性である。いずれのプログラムも入力と出力がある。不正なコードが実行されるためには、不正な入力があり、結果的に想定外の出力がおこる。JavaScript にとっての、不正な入力が起こりうる（つまり、攻撃者が自由に入力を変更できる）箇所を「ソース」とよぶ。たとえば、URL フラグメントはソースになりうる。

一方、その入力を処理した結果、不正な出力をしてしまう可能性がある。そのような箇所を「シンク」と呼ぶ。

表 3:DOM based XSS のソースとシンクの代表例

| ソース | シンク |
|---|---|
| location.hash (=URL フラグメント) | innerHTML |
| location.search | document.write |
| location.href | Eval |
| document.cookie | jQuery の html,() |
| document.referrer | jQuery の append() |
| getElementsByTagName(“input”) | jQuery の appendto() |
| jQuery の ajax()など | |

先にみてきたとおり、URL フラグメントはソースになりうる。それ以外に、jQuery の ajax()もソースになりうる。具体的には、JavaScript プログラムの開発者が安全性を保証できないファイルを ajax()で取得した場合、ファイルの中に不正コードが埋め込まれる可能性がある。(例えば、課題 4-2 では、外部 API から取得したデータの中に不正コードが入り込むかもしれない。)

他方、シンクは、ソースに渡された不正な JavaScript コードの入力が出力される関数等である。たとえば、innerHTML は、DOM ノードの HTML 記述を書き換えるものである。ここに、不正なコードが入り込むと、不正なコードが攻撃対象者のブラウザに表示され、結果的にブラウザがそのコードを実行することで、不正コードが実行されてしまう。このように、JavaScript は DOM の操作ができて便利であるが、攻撃者にコントロールされてしまうと、Web 画面の不正な書き換えを許すことも起こり得て、重大な危険性があることを承

知しておく必要がある。

○DOM based XSS の解説：JavaScript のセキュアプログラミング

これまで見てきた通り、DOM based XSS は、JavaScript の開発者にとって気を付けるべき重大な脆弱性である。しかし、コードの書き方一つで脆弱でないコードとすることができる。そのようなプログラム開発方法のことをセキュアプログラミングという。ここでは、DOM based XSS を避けるための JavaScript セキュアプログラミングの４つの原則を記載する。

１）不正な入力が入り込む可能性のある「ソース」を意識する
２）「ソース」からのデータを出力する際は、エスケープ処理・またはサニタイジング処理を行う
３）「シンク」は安全な関数を用いる
４）ライブラリを用いる場合は、最新版を利用する。

以下、２～４を説明する

２）は<script>などに含まれる特殊記号が、文字ではなく、JavaScript のコードが開始されることを示す、命令と解釈されてしまうことが原因である。そこで、ex630.html では、そのような特殊記号を無害な文字列に変換するための関数 escapeHTML() を定義している。同様の関数は、jQuery でも提供されていて、jQuery.parseHTML()がある。このような無害化を、エスケース処理またはサニタイジング処理と呼ぶ。

３）は、HTML のまま表示するから問題がおこる。「ソース」からもらったデータは HTML として処理するのではなく、テキストとして処理すればよい。たとえば、ex420.html では、jQuery セレクタで選択した DOM ノードに対して、html()メソッドで情報を書き込んでいるが、もしも、外部 API から受け取ったデータの中に不正コードが入り込んでいた場合、HTML として出力されてしまう。そこで、HTML として処理すべき部分には、「ソース」からのデータが入り込まないようにし、そのうえで text()メソッドを使うと安全になる。そのための代替関数の例を表 4 に示す。どうしても危険性が高いシンク手段を利用せざるを得ない場合は、必ず対策２によって不正コードの出力を防ぐこと。

４）を説明するために、過去の例を紹介する。2011 年 6 月 20 日にリリースされた jQuery Mobile の Beta 1 というバージョンには、ドメインが異なるサイトの HTML を読み込み、表示中のページに展開してしまう DOM based XSS 脆弱性が発見された。その後、2011 年 8 月 3 日に公開された Beta 2 で、その脆弱性が解消された。一方、利用者が古いバージョンの jQuery Mobile を利用し、なおかつ、そのバージョンの脆弱性の存在や、新バージョンのリリースを把握していなければ、脆弱な JavaScript コードを継続利用してしまうことに

なってしまう。このためバージョンアップに気を配る必要がある。

表 4: DOM based XSS に関係するシンク関数・手段

| 危険性のある関数・手段 | 安全な関数・手段 |
|---|---|
| element.innerHTML | element.textContent (テキストとして代入) |
| document.write() | 代替関数はないので、別の手段で出力 |
| eval() | 危険性が高いので、原則として利用禁止 |
| jQuery の html() | jQuery の text() (テキストとして出力) |
| jQuery の append()、appendto() | 代替関数はないので、別の手段で出力 |

**参考文献**：


(1) 安藤健一，杉本吉章，小田倉竜也著，『JavaScript Ajax・jQuery・HTML5/CSS3 のキホン』，インプレスジャパン

(2) 岡本隆史，梶原直人，田中智文著，『jQuery Mobile スマートフォンアプリ開発』，ソフトバンククリエイティブ

(3) 河村嘉之，川尻 剛著，『プロになるための JavaScript 入門』，技術評論社

(4) 大津真 著、『JavaScript&jQuery レッスンブック』、ソシム、2011 年 11 月

(5) 高津戸壮、小原司 著、『改訂版 Web デジアナーのための jQuery 入門』、技術評論社、2014 年 12 月

(6) 古籏一浩、いちがみトモロヲ、KLEE、Atelier*Spoon、錦織幸知、山本圭助、前田瑞紀、津留敏哉 著、『すべての人に知っておいてほしい jQuery & jQuery Mobile の基本原則』、エムディエヌコーポレーション、2012 年 10 月

(7) 谷口隼祐、永安佑希允 著、『DOM Based XSS に関するレポート』、2013 年 1 月、https://www.ipa.go.jp/files/000024729.pdf

(8) はせがわようすけ 著、『第 6 回 DOM-based XSS その 1』、2016 年 10 月、http://gihyo.jp/dev/serial/01/javascript-security/0006

(9) Jim Manico et al., “DOM based XSS Prevention Cheat Sheet,” Aug. 2016, https://www.owasp.org/index.php/DOM_based_XSS_Prevention_Cheat_Sheet

(10) 鈴木 富明、白井 丈晴、小林 真也、川端 秀明、西垣 正勝、『セキュリティガイドラインに準拠したアプリケーション作成支援に関する一提案』、情報処理学会・研究報告、コンピュータセキュリティ(CSEC)，Vol.2015-CSEC-68, No.8, pp.1-8, 2015 年 3 月.

(11) 徳丸浩 著、『Web セキュリティ教室』、日経ＢＰ社、2015 年 10 月


以上