ПО мобильных устройств

Лабораторная работа №2.

Разработка событийно-ориентированных мобильных приложений по технологии HTML5/CSS3/JS

Цель работы Приобретение навыков разработки мобильных приложений Cordova с использованием датчиков и сенсоров мобильного устройства.

Введение

Событи́йно-ориенти́рованное программи́рование (англ. *event-driven programming*, СОП) — парадигма программирования, в которой выполнение программы определяется событиями — действиями пользователя (клавиатура, мышь, сенсорный экран), сообщениями других программ и потоков, событиями операционной системы (например, поступлением сетевого пакета) [1].

СОП –это также способ построения программы, при котором в коде явным или неявным образом выделяется *главный цикл приложения*, тело которого состоит из двух частей: *выборки события* и *обработки события*.

Как правило, в реальных задачах оказывается недопустимым длительное выполнение обработчика события, поскольку при этом программа не может реагировать на другие события. В связи с этим при написании событийно-ориентированных программ часто применяют автоматное программирование.

Событие – это сообщение об изменении состояния объекта или системы. Так, для объектов класса "человек" событием может быть рождение или смерть, свадьба или развод.

В программировании различают системные события и события, связанные с интерфейсными объектами. Системные события генерируются («зажигаются») изменениями датчиков, интерфейсов и устройств аппаратной платформы. Примеры таких событий – разряд батареи, поворот устройства, сетевое сообщения и др. Интерфейсные события в мире программных объектов обычно связывают с действиями пользователя. Так, командная кнопка может быть нажата - событие ***Click***, документ может быть закрыт - событие ***Close***, в список может быть добавлен новый элемент – событие ***Changed***.

Разные языки программирования в разной степени поддерживают СОП. Наиболее полной поддержкой событий обладают следующие языки (неполный список): Perl, Delphi (язык программирования), C#, JavaScript.

Событийно-ориентированное программирование, как правило, применяется в трёх случаях:

1. разработка пользовательских интерфейсов (в том числе графических);
2. разработка высоконагруженных серверных приложений;
3. разработка игр, требующих управления множеством объектов.

Системные и интерфейсные программные объекты обладают предопределенными наборами/типами событий [2]. ***Тип события*** - это строка, именующая событие. Тип «mousemove», например, означает, что пользователь переместил указатель мыши. Тип «keydown» означает, что была нажата клавиша на

клавиатуре. А тип «load» означает, что завершилась загрузка документа (или другого ресурса) из сети.

Разработчик может дополнить набор событий отдельных объектов с помощью переопределения классов (создания наследников) соответствующих объектов и использования объектов переопределенных классов.

Клиентские программы на языке JavaScript основаны на СОП. Это означает, что JavaScript-приложение отслеживает определенные типы событий и может зарегистрировать одну или более функций, которая будет вызываться/исполняться при возникновении этих событий. Когда говорят о событии, обычно упоминают тип и объект, в котором возникло событие. Например, событие «load» объекта Window или событие «click» элемента <button>.

СОП приложения постоянно ждут появления событий (главный цикл приложения – просмотр и выборка из очереди событий), и выполняют обработку событий с помощью соответствующей функции.

Обработчик события - это функция, которая исполняется при возникновении («зажигании») события. Приложения должны зарегистрировать свои функции обработчиков, указав тип события и объект. Когда в указанном объекте возникнет событие указанного типа, вызовет обработчик. С одним и тем же событием может быть связано несколько обработчиков, которые будут исполняться последовательно. Многие события не имеют зарегистрированных обработчиков – такие события приложение игнорирует (не обрабатывает).

При возникновении некоторых событий объекты-источники могут передавать обработчикам дополнительную информацию о событии в форме объекта-аргумента функции обработчика. Например, объект, связанный с событиями от мыши, включает координаты указателя мыши, а объект, связанный с событиями от клавиатуры, содержит информацию о нажатой клавише и клавишах-модификаторах. Для многих типов событий определяются только стандартные свойства type и target. Для таких типов событий важен только факт события, и никакая другая информация не имеет значения.

Порядок программирования СОП приложений JavaScript

Политика безопасности Cordova не допускает встроенных событий в файлах ****.html***, поэтому весь JavaScript-код, в частности функции-обработчики событий, должны размещаться в файлах ***JS/*.js***, например ***index.js***.

При разработке СОП мобильных приложений на языке JavaScript следует придерживаться следующей последовательности действий.

1. Определить объекты-источники необходимых для работы приложения событий.

Примерами системных источников событий являются батарея питания, интерфейс вычислительной или телефонной связи, измерительные датчики различных физических величин. Для формирования программных объектов таких источников в технологии HTML/JS обычно необходимо использовать соответствующий плагин сторонних разработчиков. Список плагинов можно найти, например, в [3]. Каждый плагин обеспечивает объект для программного взаимодействия обычно с одним аппаратным устройством. Для такого объекта также необходимо иметь список имен событий, методов и описания информационных объектов. Следует также учитывать, что плагин может быть платформозависимым.

Список доступных плагинов и их описания можно найти по адресу [3].

Если существующие в репозитариях плагины не обеспечивают требуемой функциональности, то единственный выбор — создать плагин самостоятельно.

Примерами интерфейсных источников событий являются кнопки, поля ввода и др. Эти элементы формируют модель DOM приложения. Для доступа к DOM-объектам в HTML-коде такие элементы следует определить с опцией ***Id="Имя_элемента"***. Например

```
<button Id="TestButton">Test</button>
```

Тогда ссылку на элемент в JavaScript-коде можно получить с помощью функции

```
var button = document.getElementById('Имя_элемента')
```

Например

```
var testBtn = document.getElementById("TestButton")
```

2. Зарегистрировать функцию-обработчик необходимого типа события

   Регистрацию обработчика выполняют с помощью функции

```
<Объект>.addEventListener(<Тип _события>, <Обработчик>, [Options]);
```

Например, код

```
testBtn.addEventListener("click",onClick);
```

обеспечит вызов функции onClick при нажатии на кнопку Test.

3. Создать функцию- обработчик

   Функция ***onClick()***, указанная в примере, должна быть определена в соответствии с требуемой реакцией приложения на нажатие кнопки. Например, известить пользователя о нажатии кнопки

```
function onClick() {
   alert('Test button is Pressed!');
}
```

Системные события

В гибридных проектах Cordova разработчик может использовать различные системные типы событий. Основные из них перечислены в следующей таблице.

Тип события	**Описание**
deviceready	**Приложение загружено, устройство готово к работе.** Это событие «зажигается» после завершения полной загрузки приложения Cordova. Событие гарантирует, что указанные в приложении ресурсы загружены. Используется для гарантии, что никакие функции Cordova не будут вызваны до полного завершения размещения приложения в памяти.
pause	**Пауза** Это событие «зажигается» при переходе приложение в фоновый режим работы, например, после нажатия на телефоне кнопки «домой».
resume	**Продолжить**

	Это событие «зажигается», когда приложение возвращается из фонового режима.
backbutton	**Кнопка НАЗАД** Это событие «зажигается» при нажатии кнопки «Назад».

menubutton	**Кнопка MENU (Последние приложения)**
	Это событие вызывается при нажатии кнопки меню.
	
	Начиная с Android 3.0 (уровень API 11), устройства больше не обязаны иметь системную выделенную кнопку меню. Для эмуляции кнопки меню в Android необходимо добавить строку, разрешающую переопределение обработчика кнопки «Последние приложения»:
	
	`navigator.app.overrideButton("menubutton", true);`
	
	перед назначение обработчика
	
	`document.addEventListener("menubutton", onMenuKeyDown, false);`

Примеры разработки приложений с Apache Cordova:
https://coderlessons.com/tutorials/mobilnaia-razrabotka/vyuchit-kordovu/kordova-kratkoe-rukovodstvo

Порядок выполнения работы

1. Ознакомьтесь с принципами СОП и порядком их разработки.
2. Выполните разработку и отладку приложений, указанных в работе.
3. Проведите анализ кода этих приложений.
4. Оформите отчет

Задания для самостоятельной работы

1. Создайте приложение, обеспечивающее вывод на экран показаний датчика в соответствии с заданным номером варианта:

№	Датчик
1	Акселерометр
2	Магнитный компас
3	Освещенность
4	Гироскоп
5	Барометр
6	Геолокация
7	Освещенность
8	Приближения
9	Барометр

10	Температура

Опишите установленные плагины, принципы работы приложения.

2. Скопируйте приложение из п.1 Задания, переименуйте его и добавьте функциональности. Например, вывод сообщений о достижении контролируемых значений из заданных пределов.

Контрольные вопросы

1. Что такое СОП? Опишите принципы СОП.
2. Что в программировании называется событием?
3. Какие виды объектов являются источниками событий?
4. Что такое «тип события»?
5. Что такое «обработчик события»?
6. Какова последовательность разработки СОП приложения по технологии HTML5/JS в среде Cordova?
7. Какова роль плагинов в приложениях HTML5/JS?

Литература

1. https://ru.wikipedia.org/wiki/Событи́йно-ориенти́рованное программи́рование
2. https://professorweb.ru/my/javascript/js_theory/level2/2_6.php - Типы событий.
3. http://phonegap-plugins.com/ - список плагинов