# Developing Tizen Web Applications

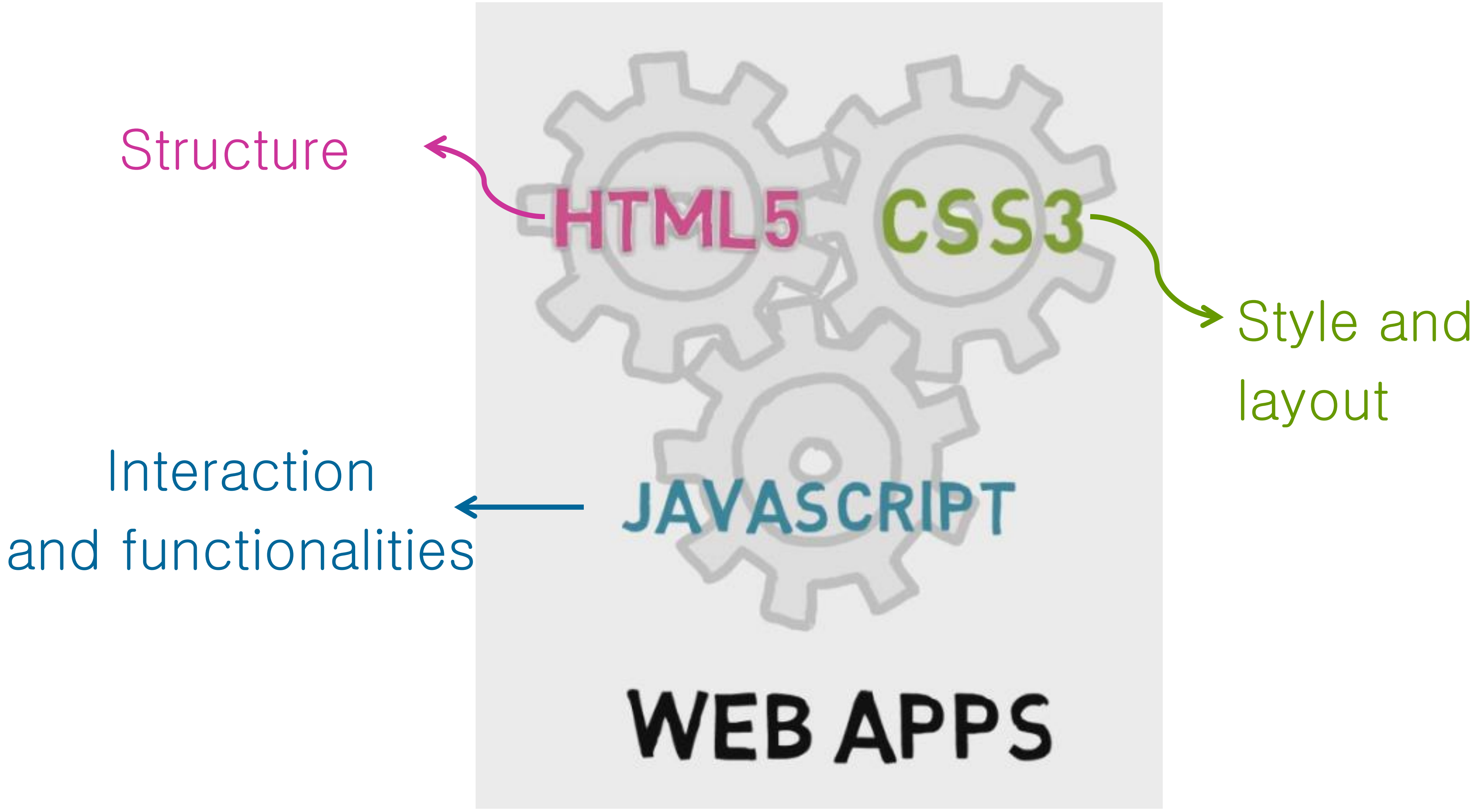
* This document is based on Tizen Studio

# Table of Contents

## Tizen Web Applications Overview

## Introduction

Web applications involve the standards for building and rendering web pages, such as HTML, CSS, and JavaScript.



# Introduction

Web applications offer a range of benefits. It is a perfect way to start developing Tizen applications.

**Benefits of Web Applications**

- Easy to learn
- Compatible across multiple mobile and wearable platforms
- Easy to maintain

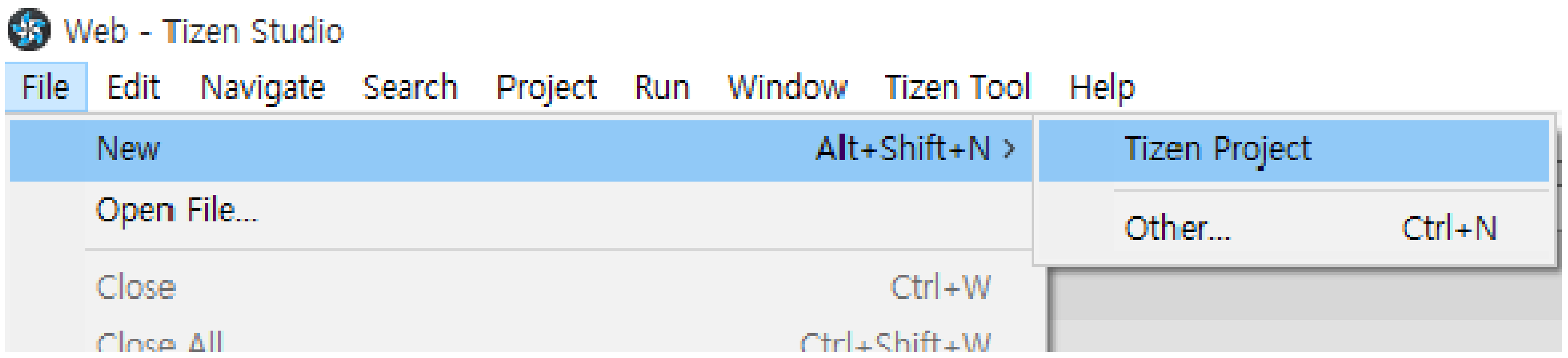
However, there are still some limitations of Web applications over native applications. Creation of native applications is discussed later in this tutorial.

**Limitations of Web Applications**

- Relatively low performance
- Limited access to the mobile and wearable device's features

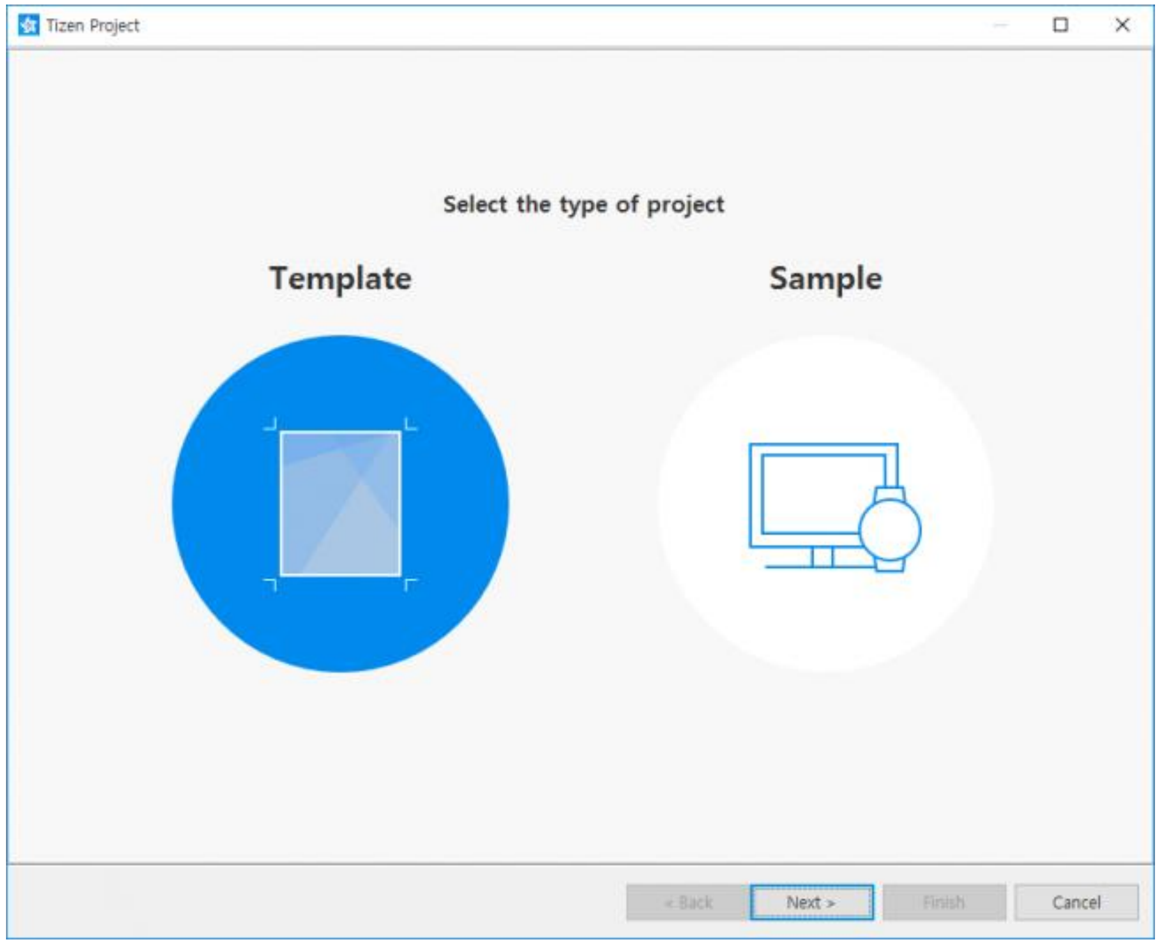
## Creating Tizen Web Project

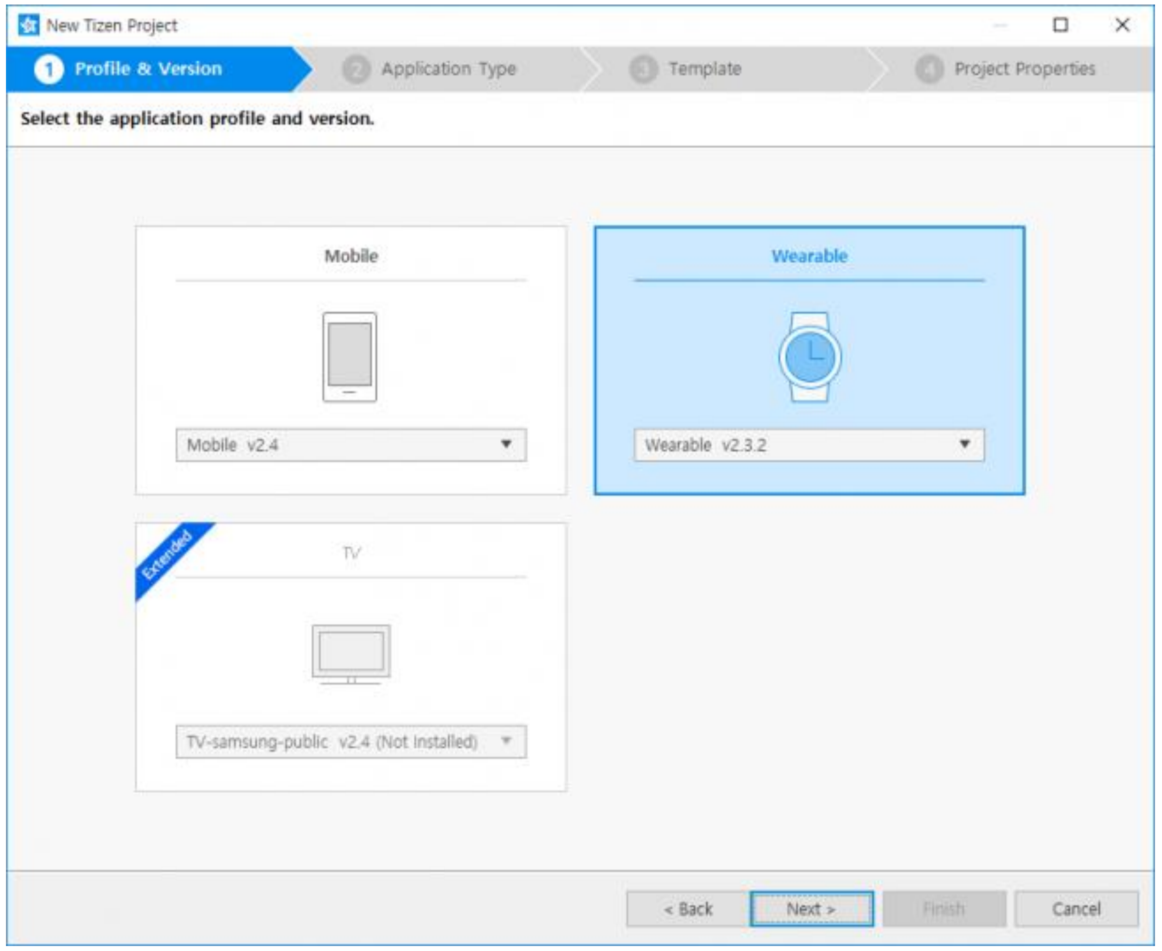
In the Tizen IDE, click **File > New > Tizen Project**.



# Creating Tizen Web Project

In the project wizard pop-up window, select the **Template**, then select profile and versions: **WEARABLE v2.3.2** application.





Select the application profile and version.

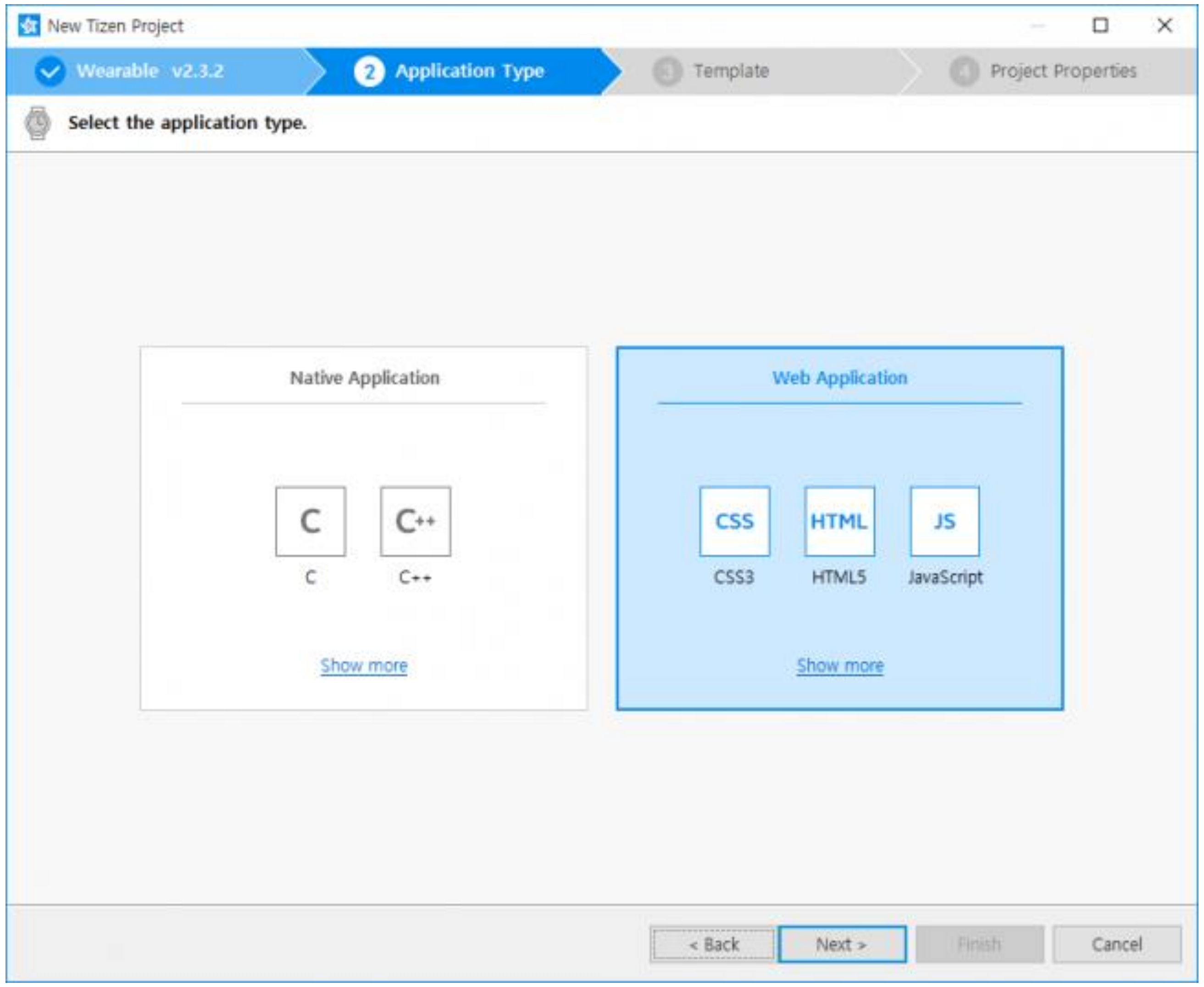
Mobile v2.4

Wearable v2.3.2

TV-samsung-public v2.4 (Not Installed)

## Creating Tizen Web Project

Select Web Application and click Next. Then select the application template.

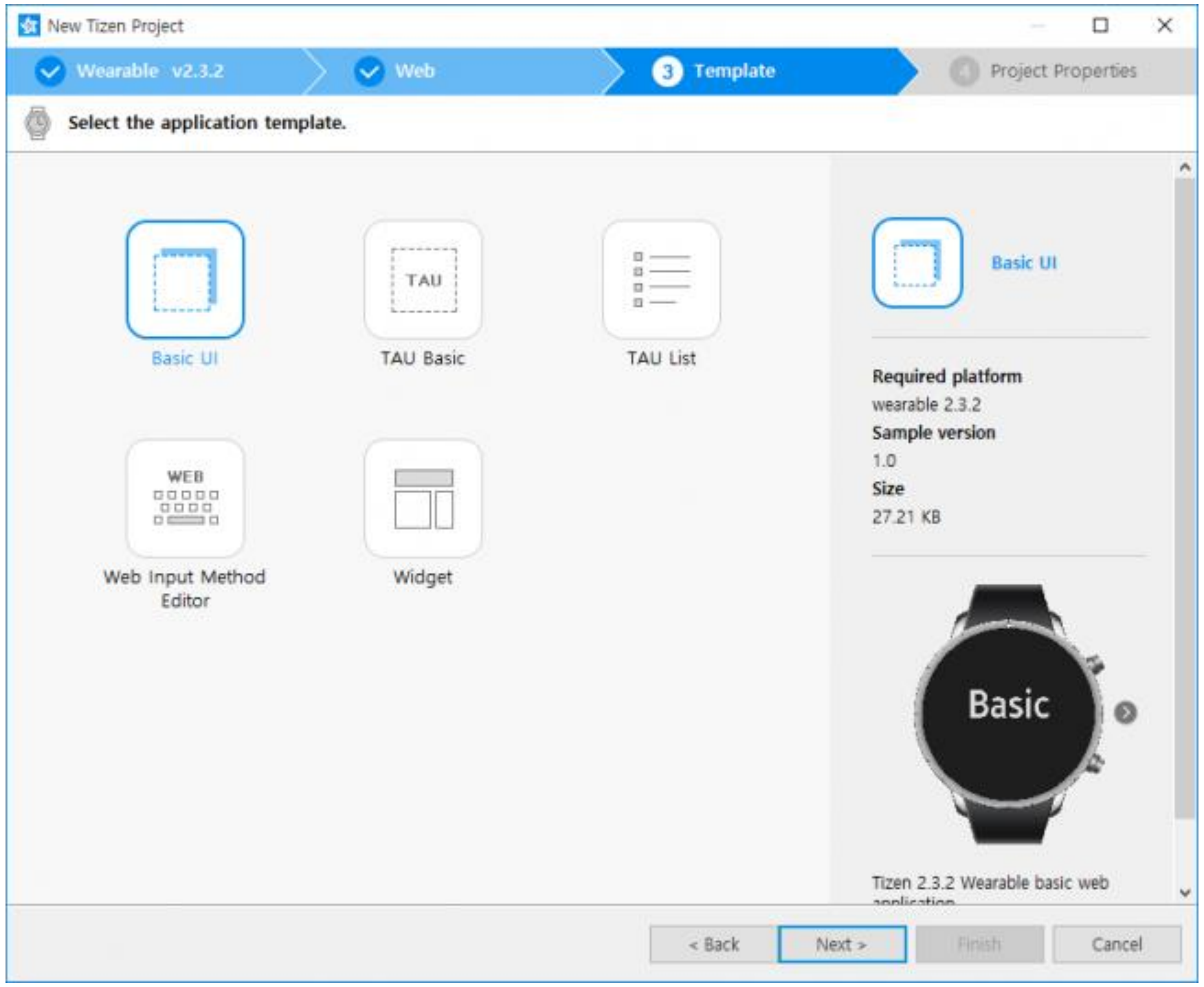

New Tizen Project

Wearable v2.3.2 → 2 Application Type → 3 Template → 4 Project Properties

Select the application type.

| Native Application | Web Application |
|---|---|
| C, C++ | CSS3, HTML5, JavaScript |
| Show more | Show more |

< Back | Next > | Finish | Cancel


New Tizen Project

Wearable v2.3.2 → Web → 3 Template → 4 Project Properties

Select the application template.

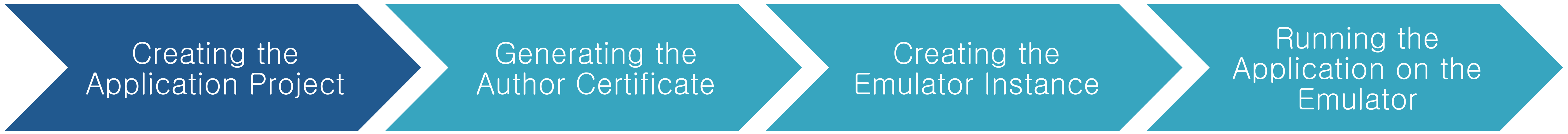
Basic UI, TAU Basic, TAU List, Web Input Method Editor, Widget

Basic UI

Required platform
wearable 2.3.2
Sample version
1.0
Size
27.21 KB

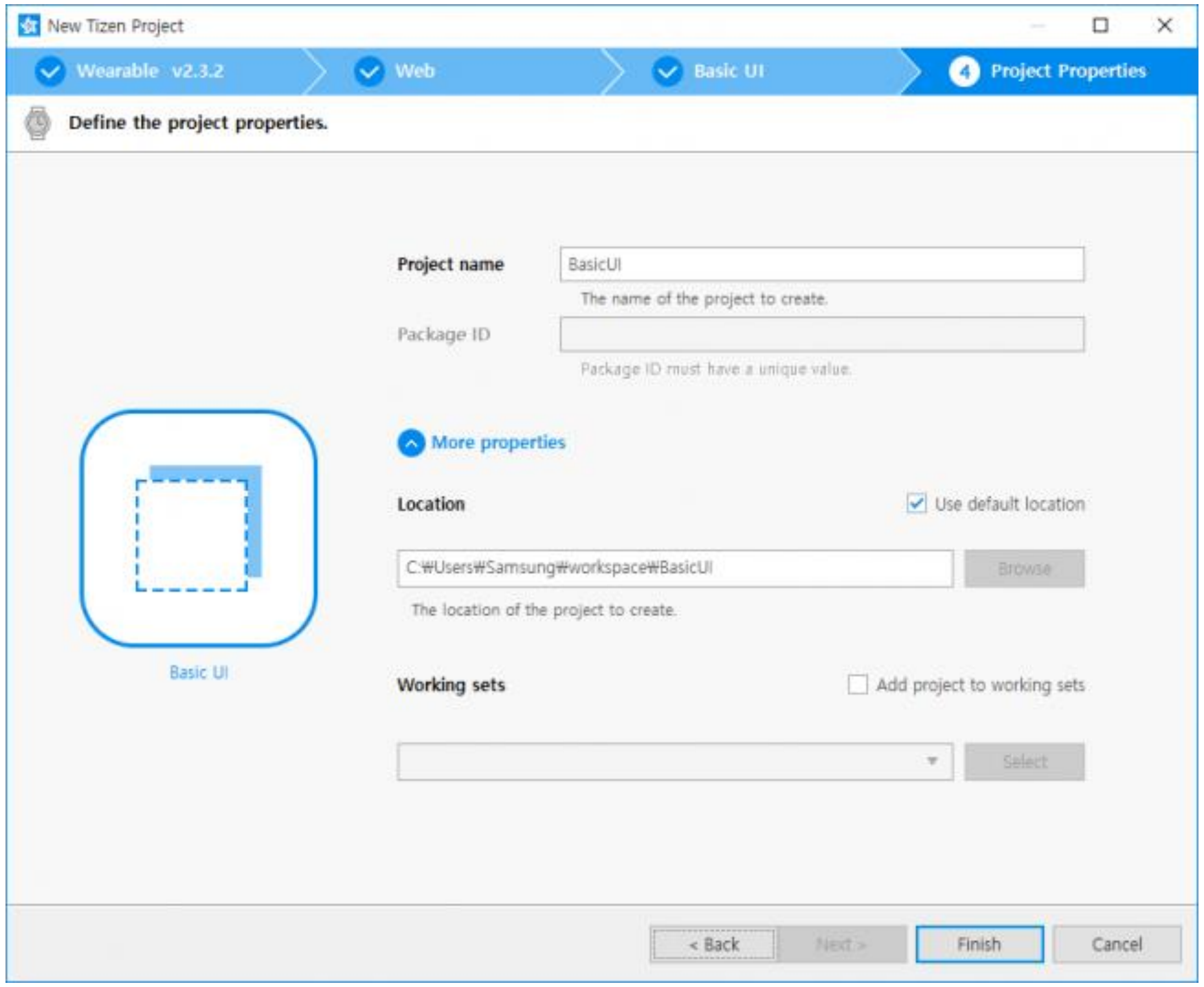
Tizen 2.3.2 Wearable basic web application

< Back | Next > | Finish | Cancel


Creating the Application Project
Generating the Author Certificate
Creating the Emulator Instance
Running the Application on the Emulator

# Creating Tizen Web Project

Change the project name, if you want.
Click More properties, if you want to change more properties.
Leave all other fields in the Project Wizard to their default values, and click Finish.



New Tizen Project

Wearable v2.3.2 > Web > Basic UI > 4 Project Properties

Define the project properties.

Project name: BasicUI
The name of the project to create.

Package ID:
Package ID must have a unique value.

More properties

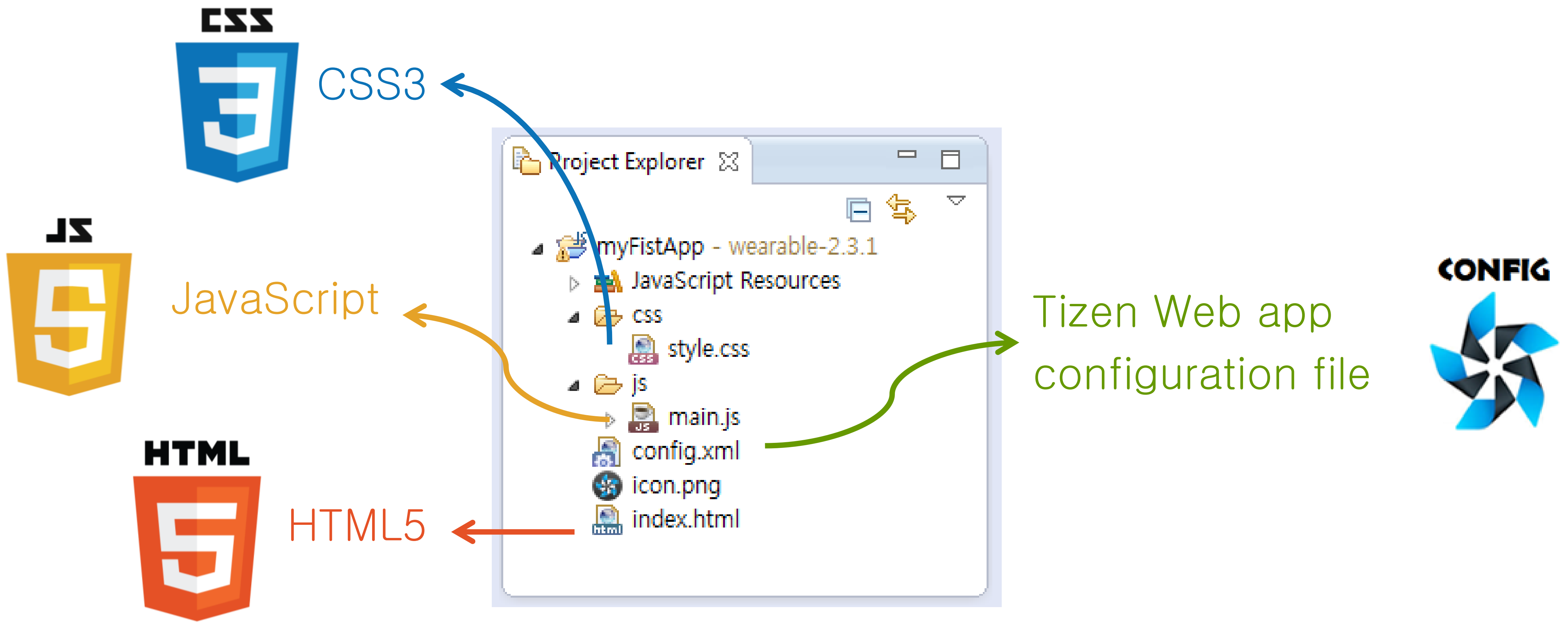
Location — ✓ Use default location
C:₩Users₩Samsung₩workspace₩BasicUI
The location of the project to create.

Working sets — ☐ Add project to working sets

Creating the Application Project
Generating the Author Certificate
Creating the Emulator Instance
Running the Application on the Emulator

## Files in Tizen Web Project

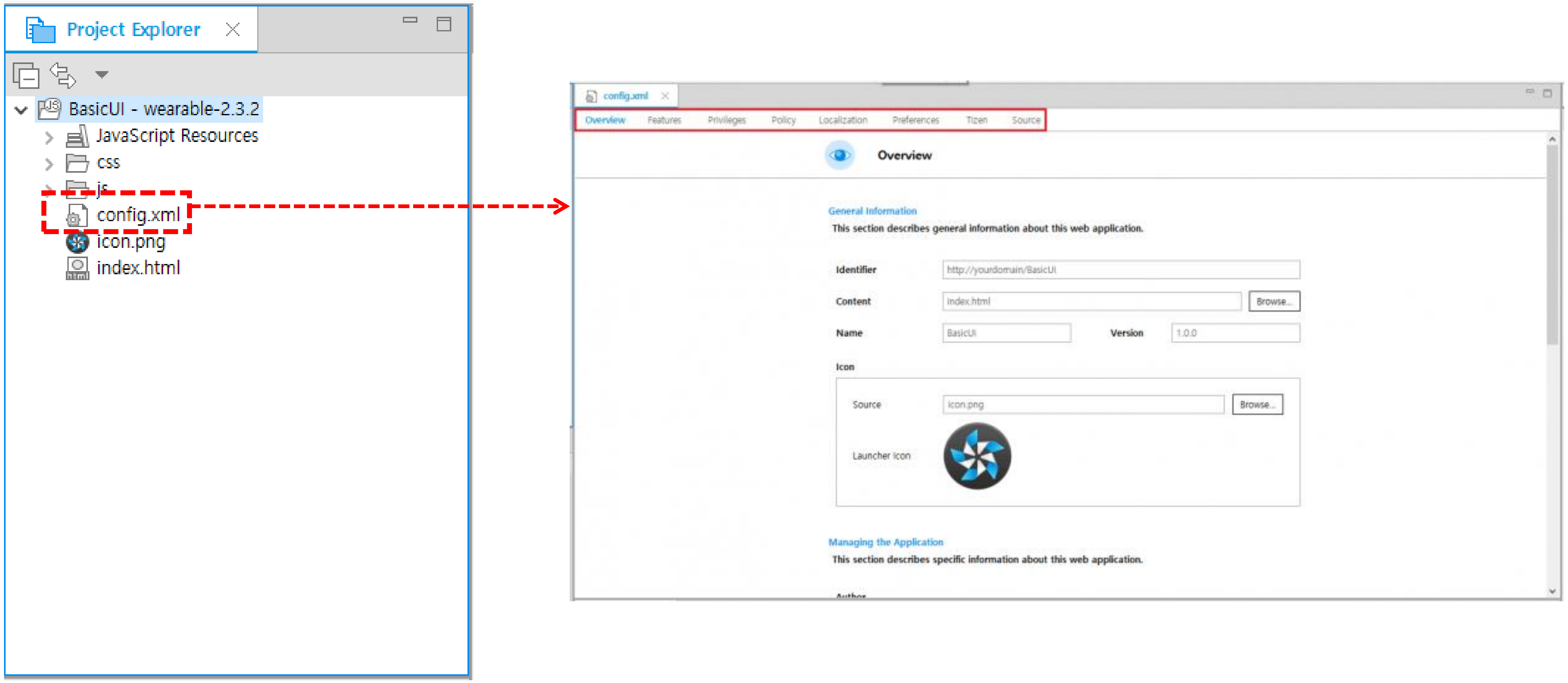
The following figure illustrates the basic structure of a Tizen Web application.



The following pages describe each file type in detail.

## Files in Tizen Web Project

The `config.xml` file is a Web-standard deployment descriptor for the Web application. It defines everything about the application that is required when the application is installed and launched.



General Information

This section describes general information about this web application.

| | |
|---|---|
| Identifier | http://yourdomain/BasicUI |
| Content | index.html |
| Name | BasicUI |
| Version | 1.0.0 |

Icon

| | |
|---|---|
| Source | icon.png |
| Launcher Icon | |

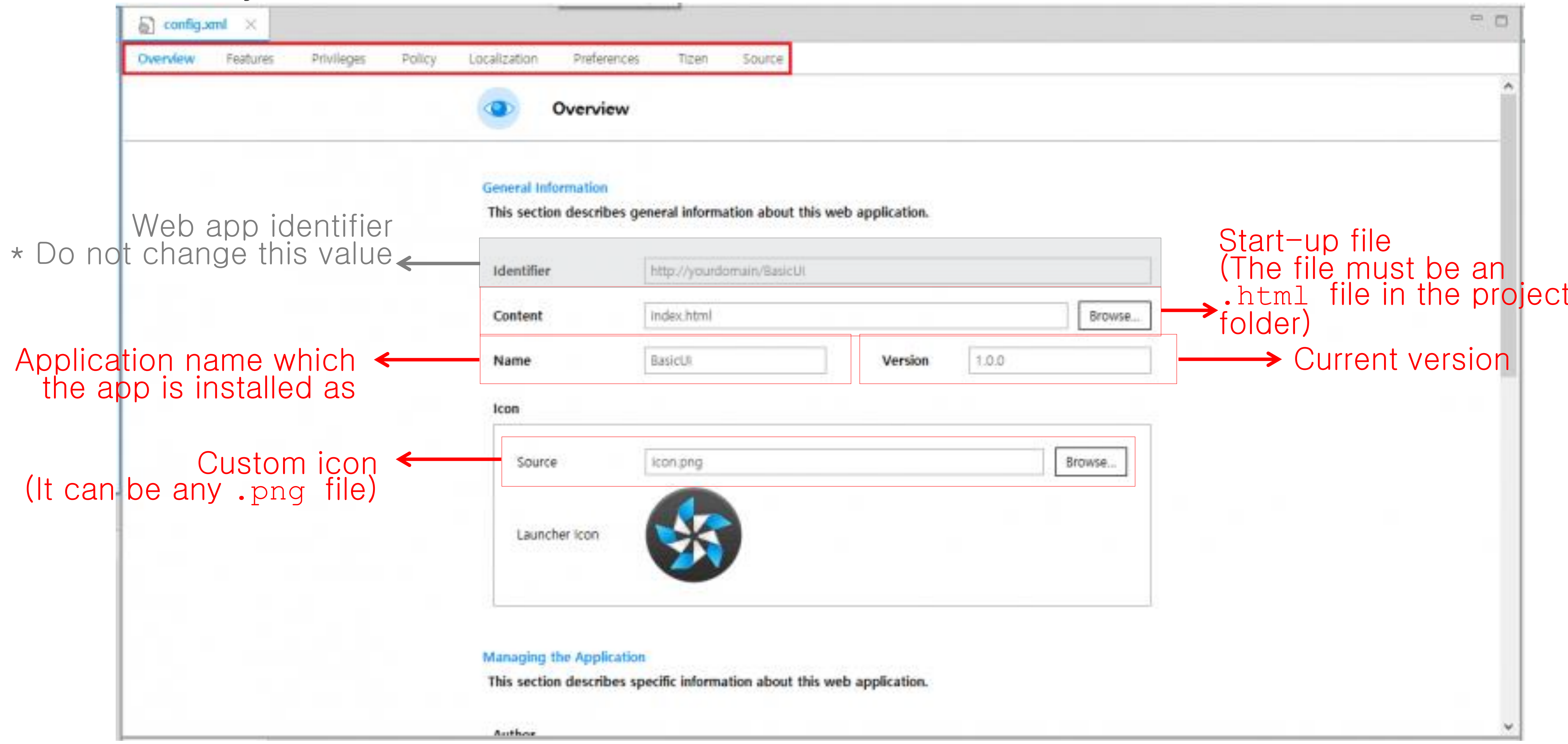
Managing the Application

This section describes specific information about this web application.

Author

# Files in Tizen Web Project

The Tizen SDK automatically fills in the general information with default values, and you can change the values using the configuration editor if necessary.



config.xml

Overview
Features
Privileges
Policy
Localization
Preferences
Tizen
Source

Overview

General Information

This section describes general information about this web application.

| | |
|---|---|
| Identifier | http://yourdomain/BasicUI |
| Content | index.html |
| Name | BasicUI |
| Version | 1.0.0 |

Icon

| | |
|---|---|
| Source | icon.png |
| Launcher icon | |

Managing the Application

This section describes specific information about this web application.

Author

Browse...
Web app identifier
* Do not change this value
Start-up file
(The file must be an .html file in the project folder)
Application name which the app is installed as
Current version
Custom icon
(It can be any .png file)

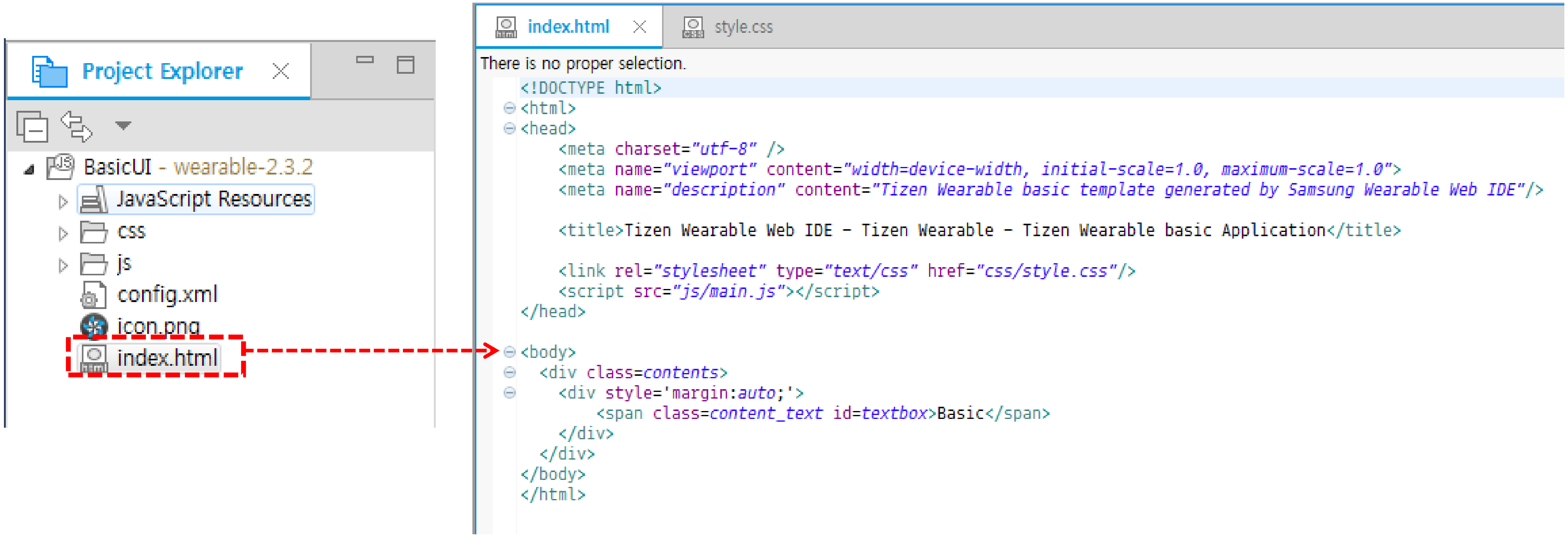
## Files in Tizen Web Project

You can set the application configuration with various options in each tab.

- **Overview**: Define and edit general information, such as the application name and icon.
- **Features**: Declare the required software and hardware features.
- **Privileges**: Specify the APIs or API groups accessed and used.
- **Policy**: Request network resource permissions when required to access external network resources.
- **Localization**: Provide localization support for name, description, and license elements of the `config.xml` file.
- **Preferences**: Declare the name-value pairs which can be set or retrieved.
- **Tizen**: Edit the Tizen schema extension properties.
- **Source**: View and edit the code of the `config.xml` file.

# Files in Tizen Web Project

The `index.html` file contains source codes for the structure of the application view.



There is no proper selection.

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0, maximum-scale=1.0">
    <meta name="description" content="Tizen Wearable basic template generated by Samsung Wearable Web IDE"/>

    <title>Tizen Wearable Web IDE - Tizen Wearable - Tizen Wearable basic Application</title>

    <link rel="stylesheet" type="text/css" href="css/style.css"/>
    <script src="js/main.js"></script>
</head>

<body>
  <div class=contents>
    <div style='margin:auto;'>
        <span class=content_text id=textbox>Basic</span>
    </div>
  </div>
</body>
</html>
```

# Files in Tizen Web Project

**H**T**ML** stands for **H**yper **T**ext **M**arkup **L**anguage, and the HTML documents are made up of HTML elements, which are written with a **start tag**, an **end tag**, and **contents** in between.

```
<tagname>contents</tagname>
```

Everything from the start tag to the end tag is called an HTML **element**.

HTML elements can have **attributes**, which provide additional information about the elements.

```
<tagname attributename=“value”>contents</tagname>
```

The list of standard HTML elements is provided by the W3C Working Group:
https://www.w3.org/TR/html-markup/elements-by-function.html

# Files in Tizen Web Project

The following example explains the elements in a Web application's `index.html` file:

`index.html`

```html
<!DOCTYPE html>
<html>

<head>
    <meta charset="utf-8" />
    <meta name="viewport"
          content="width=device-width, initial-
          scale=1.0, maximum-scale=1.0">
    <meta name="description"
          content="Tizen Wearable Web Basic Template" />

    <title>Tizen Wearable Web Basic Application</title>

    <link rel="stylesheet" type="text/css"
          href="css/style.css" />
    <script src="js/main.js"></script>
</head>

<body>
    <div id="main" class="page">
        <div class="contents">
            <span id="content-text">Basic</span>
        </div>
    </div>
</body>

</html>
```

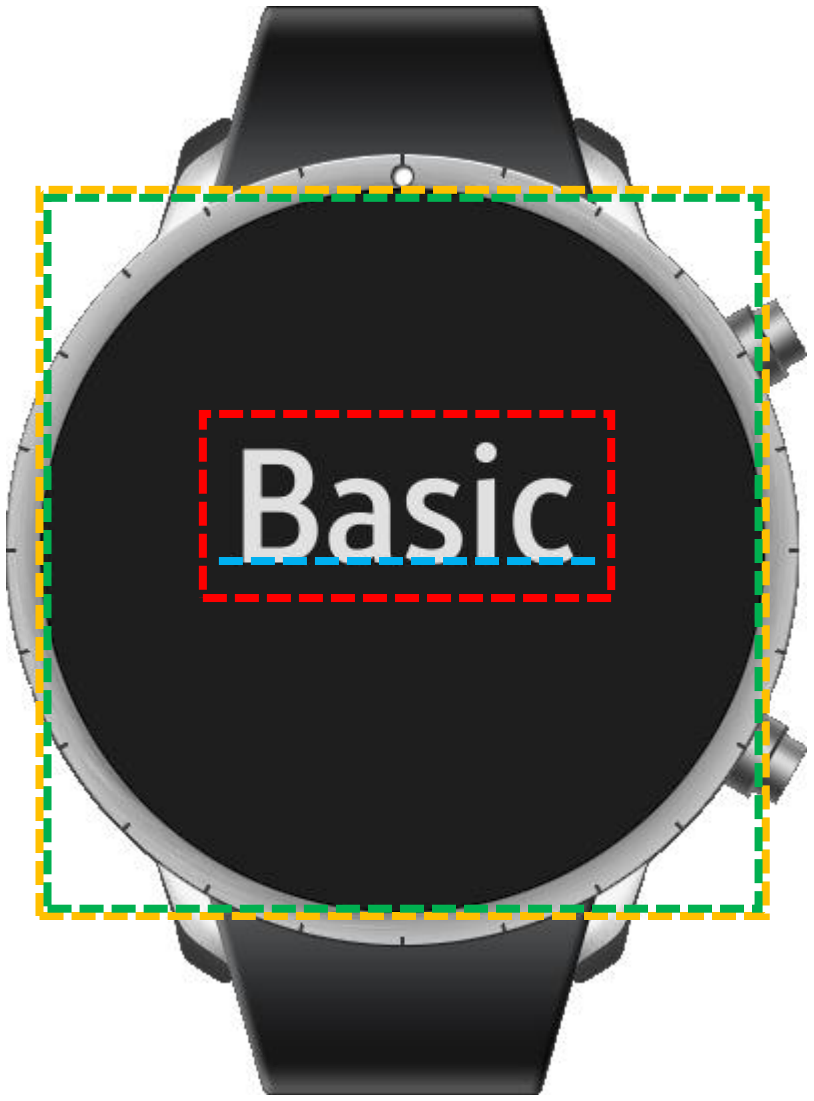
- The `DOCTYPE` declaration defines the document type to be HTML.
- The text between `<html>` and `</html>` describes the entire HTML document
- The text between `<head>` and `</head>` provides information about the document.
- The text between `<body>` and `</body>` describes the visible page content.

## Files in Tizen Web Project

The HTML code on the left represents the view on the right correspondingly.

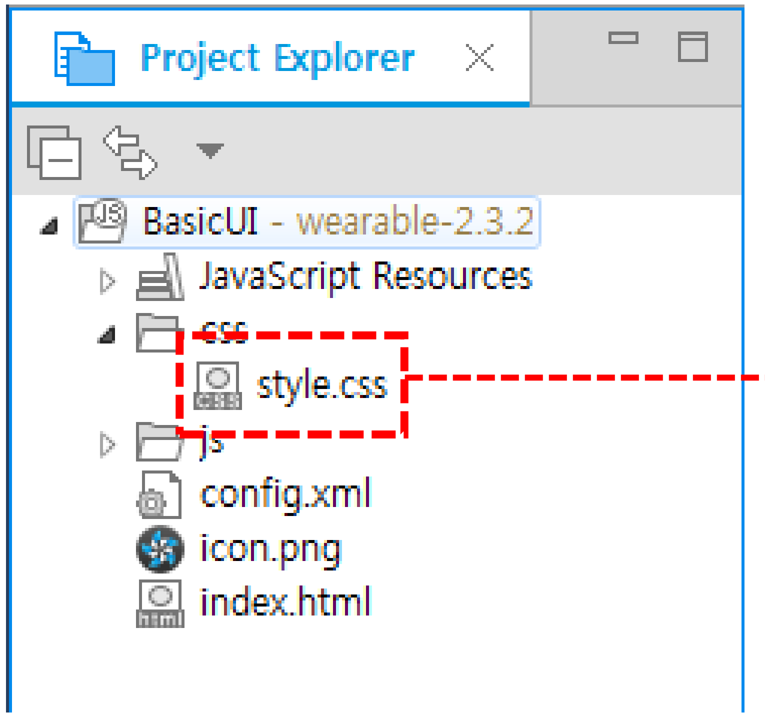
index.html

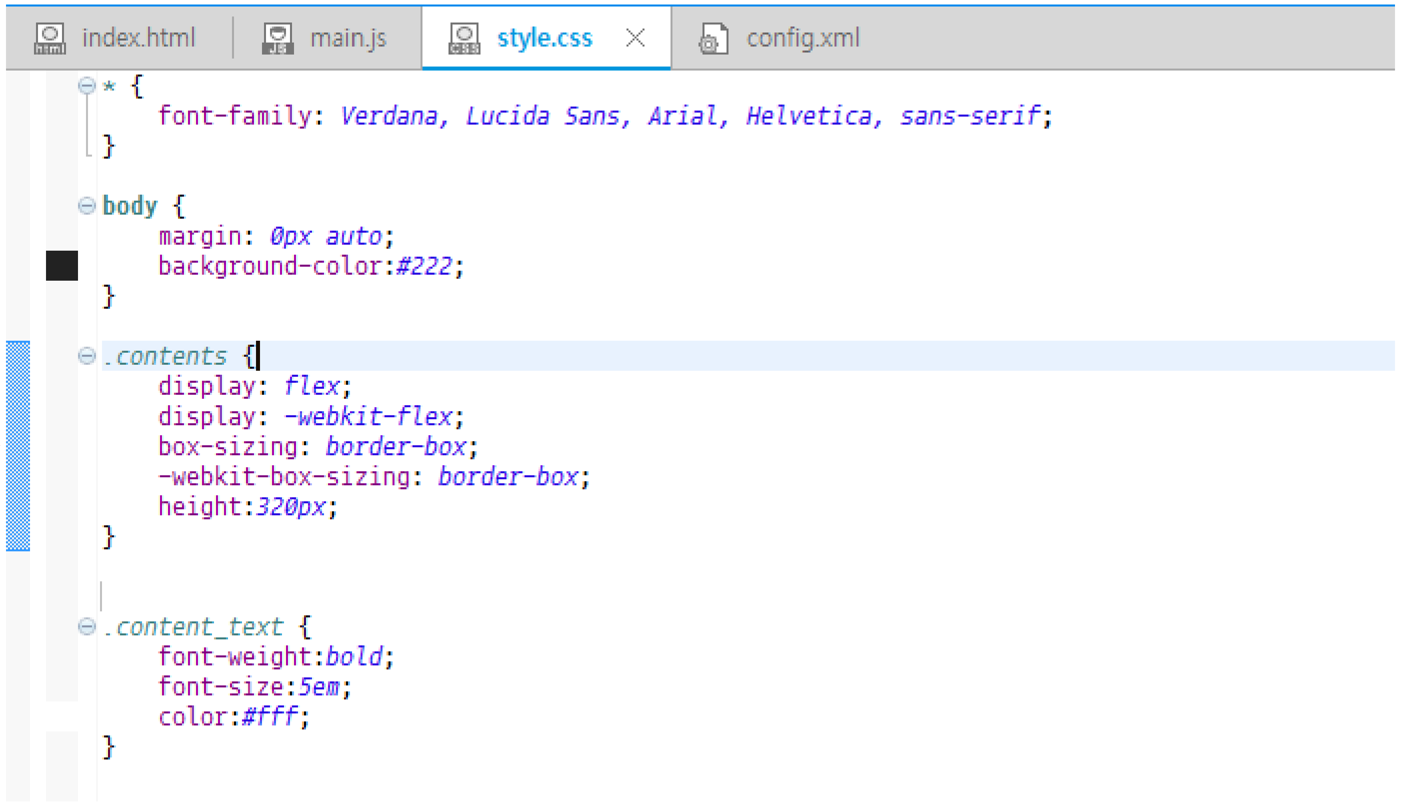
```
<!DOCTYPE html>
<html>

<head>
    <meta charset="utf-8" />
    <meta name="viewport"
          content="width=device-width, initial-
          scale=1.0, maximum-scale=1.0">
    <meta name="description"
          content="Tizen Wearable Web Basic Template" />

    <title>Tizen Wearable Web Basic Application</title>

    <link rel="stylesheet" type="text/css"
          href="css/style.css" />
    <script src="js/main.js"></script>
</head>

<body>
    <div id="main" class="page">
        <div class="contents">
            <span id="content-text">Basic</span>
        </div>
    </div>
</body>

</html>
```



# Files in Tizen Web Project

The `css/style.css` file contains the source code for specifying the layout and the styling of the application views.





```css
* {
    font-family: Verdana, Lucida Sans, Arial, Helvetica, sans-serif;
}

body {
    margin: 0px auto;
    background-color:#222;
}

.contents {
    display: flex;
    display: -webkit-flex;
    box-sizing: border-box;
    -webkit-box-sizing: border-box;
    height:320px;
}


.content_text {
    font-weight:bold;
    font-size:5em;
    color:#fff;
}
```

# Files in Tizen Web Project

The CSS file is connected to the HTML file using a `<link>` tag in the `<head>` element.

index.html

```
<!DOCTYPE html>
<html>

<head>
...

    <link rel="stylesheet" type="text/css"
          href="css/style.css" />
    <script src="js/main.js"></script>
</head>

<body>
...
</body>

</html>
```

css/style.css

```
html,
body {
    width: 100%;
    height: 100%;
    margin: 0 auto;
    padding: 0;
    background-color: #222222;
    color: #ffffff;
}
.page {
    width: 100%;
    height: 100%;
    display: table;
}
.contents {
    display: table-cell;
    vertical-align: middle;
    text-align: center;
    -webkit-tap-highlight-color: transparent;
}
#content-text {
    font-weight: bold;
    font-size: 5em;
}
```

## Files in Tizen Web Project

**C**SS stands for **C**ascading **S**tyle **S**heets, and it describes how HTML elements are displayed on the screen.

The CSS rule-set consists of a selector and a declaration block.

```
selector {
    property: value; /* declaration */
}
```

CSS selectors are used to select HTML elements based on the elements' name, ID, class, attribute, and more:

- The **element** selectors are written written as `element_name {}`.
- The **ID** selectors are written as `#id_name {}` .
- The **class** selectors are written as `.class_name {}`.

The list of the standard HTML elements is provided in the W3C Wiki page:
https://www.w3.org/community/webed/wiki/CSS/Properties

## Files in Tizen Web Project

The styling of the HTML elements in the `index.html` file is described in the `style.css` file correspondingly.

index.html

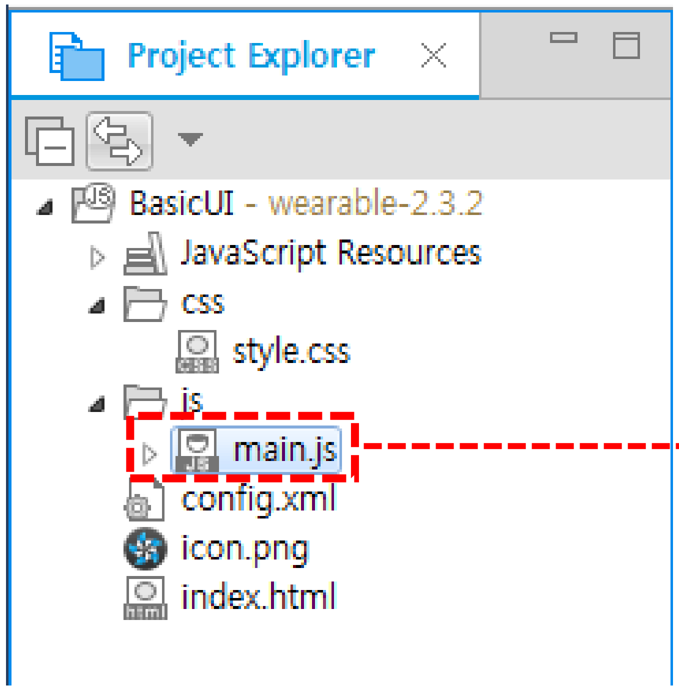
```
<!DOCTYPE html>
<html>

<head>
...

    <link rel="stylesheet" type="text/css"
href="css/style.css" />
    <script src="js/main.js"></script>
</head>

<body>
    <div id="main" class="page">
        <div class="contents">
            <span id="content-
                     text">Basic</span>
        </div>
    </div>
</body>

</html>
```

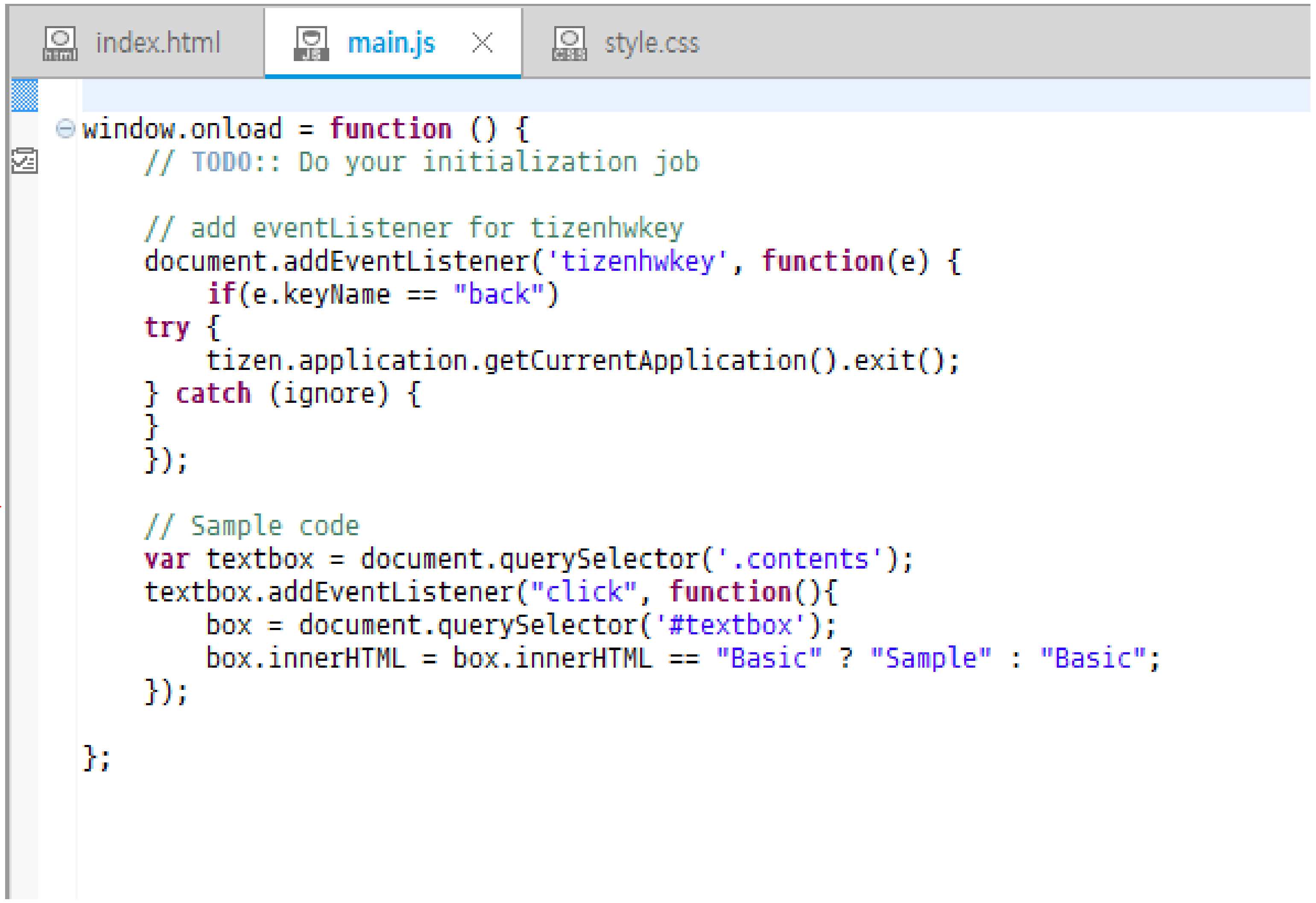
css/style.css

```
html,
body {
    width: 100%;
    height: 100%;
    margin: 0 auto;
    padding: 0;
    background-color: #222222;
    color: #ffffff;
}
.page {
    width: 100%;
    height: 100%;
    display: table;
}
.contents {
    display: table-cell;
    vertical-align: middle;
    text-align: center;
    -webkit-tap-highlight-color:
transparent;
}
#content-text {
    font-weight: bold;
    font-size: 5em;
}
```

# Files in Tizen Web Project

The `js/main.js` file contains the code for handling the application functionalities.





```
window.onload = function () {
    // TODO:: Do your initialization job

    // add eventListener for tizenhwkey
    document.addEventListener('tizenhwkey', function(e) {
        if(e.keyName == "back")
    try {
        tizen.application.getCurrentApplication().exit();
    } catch (ignore) {
    }
    });

    // Sample code
    var textbox = document.querySelector('.contents');
    textbox.addEventListener("click", function(){
        box = document.querySelector('#textbox');
        box.innerHTML = box.innerHTML == "Basic" ? "Sample" : "Basic";
    });

};
```

# Files in Tizen Web Project

The JavaScript file is connected to the HTML file using a `<script>` tag in the `<head>` element.

index.html

```
<!DOCTYPE html>
<html>

<head>
...

    <link rel="stylesheet"
          type="text/css"
          href="css/style.css" />
    <script src="js/main.js"></script>
</head>

<body>
...
</body>

</html>
```

js/main.js

```
window.onload = function() {
    // TODO:: Do your initialization job

    // Add eventListener for tizenhwkey
    document.addEventListener('tizenhwkey',
function(e) {
        if (e.keyName === "back") {
            try {

tizen.application.getCurrentApplication().exit();
            } catch (ignore) {}
        }
    });

    // Sample code
    var mainPage = document.querySelector('#main');

    mainPage.addEventListener("click", function() {
        var contentText =
document.querySelector('#content-text');

        contentText.innerHTML =
(contentText.innerHTML === "Basic") ? "Tizen" :
"Basic";
    });
};
```

## Files in Tizen Web Project

**JavaScript** is a programming language, and consists of **statements** to be executed by the program.

JavaScript statements, which are separated by **semicolons**, are typically composed of **values**, **operators**, **expressions**, **keywords**, and **comments**.

JavaScript statements to be executed together can be grouped in code blocks with curly brackets {…}, and the code blocks are called JavaScript **functions**.

```
// Sample code
var mainPage = document.querySelector('#main');

    mainPage.addEventListener("click", function() {
       var contentText = document.querySelector('#content-text');

           contentText.innerHTML = (contentText.innerHTML === "Basic") ?
           "Tizen" : "Basic";
        });
```

## Files in Tizen Web Project

With JavaScript, you can:

- Add behavior and user interactions
- Load and change contents dynamically
- Get access to device-specific features using standard Web APIs or Tizen Web Device APIs.

For example, the following JavaScript code changes the contents of the `<span>` element using the standard Web API method `element.innerHTM`.

`index.html`

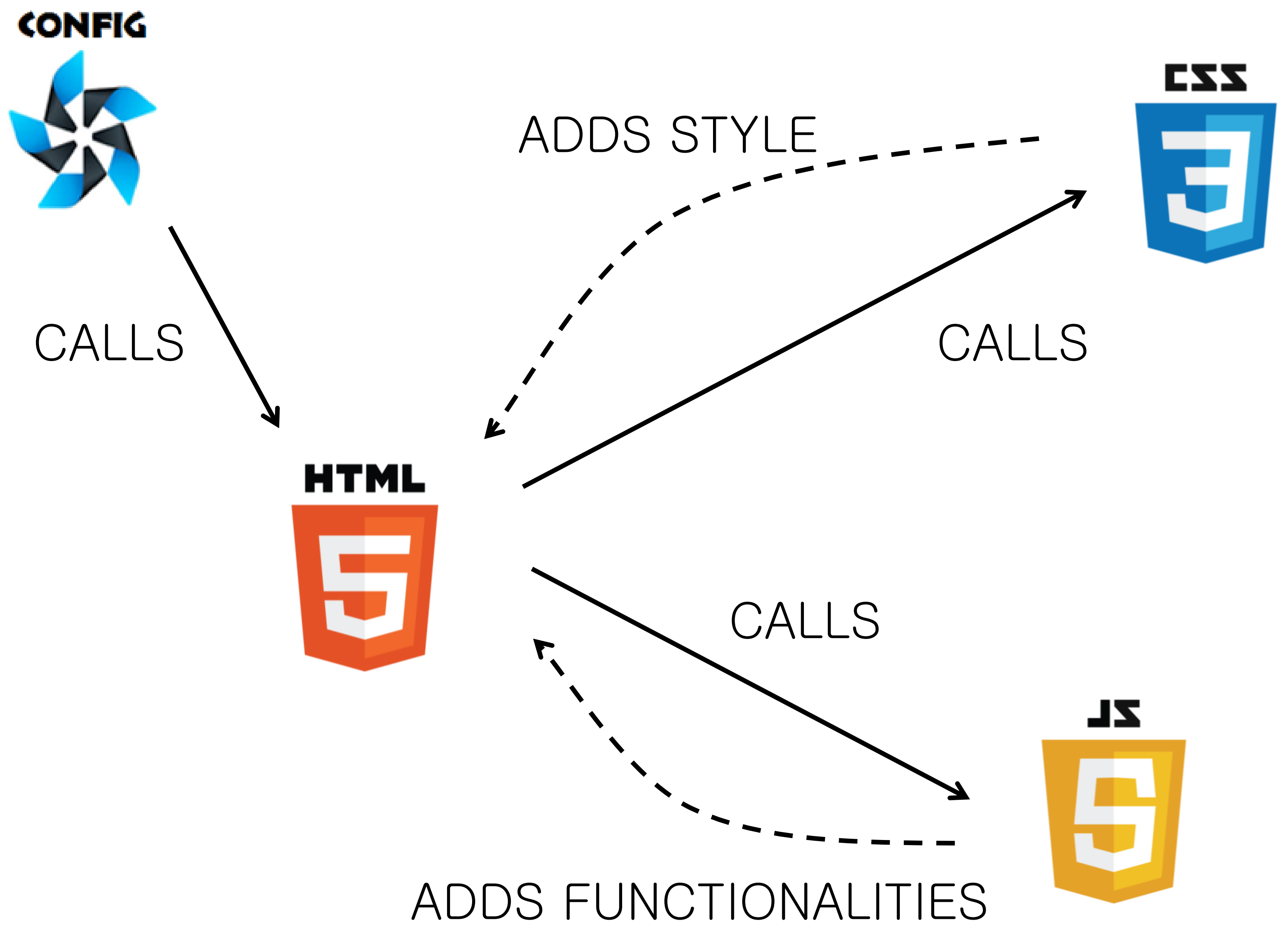
```
<body>
    <div id="main" class="page">
        <div class="contents">
            <span id="content-
                      text">Basic</span>
        </div>
    </div>
</body>
```

`js/main.js`

```
// Sample code
    var mainPage = document.querySelector('#main');
        mainPage.addEventListener("click", function()
    {
        var contentText =
            document.querySelector('#content-text');
            contentText.innerHTML =
            (contentText.innerHTML === "Basic") ?
            "Tizen" : "Basic";
    });
```

## Files in Tizen Web Project

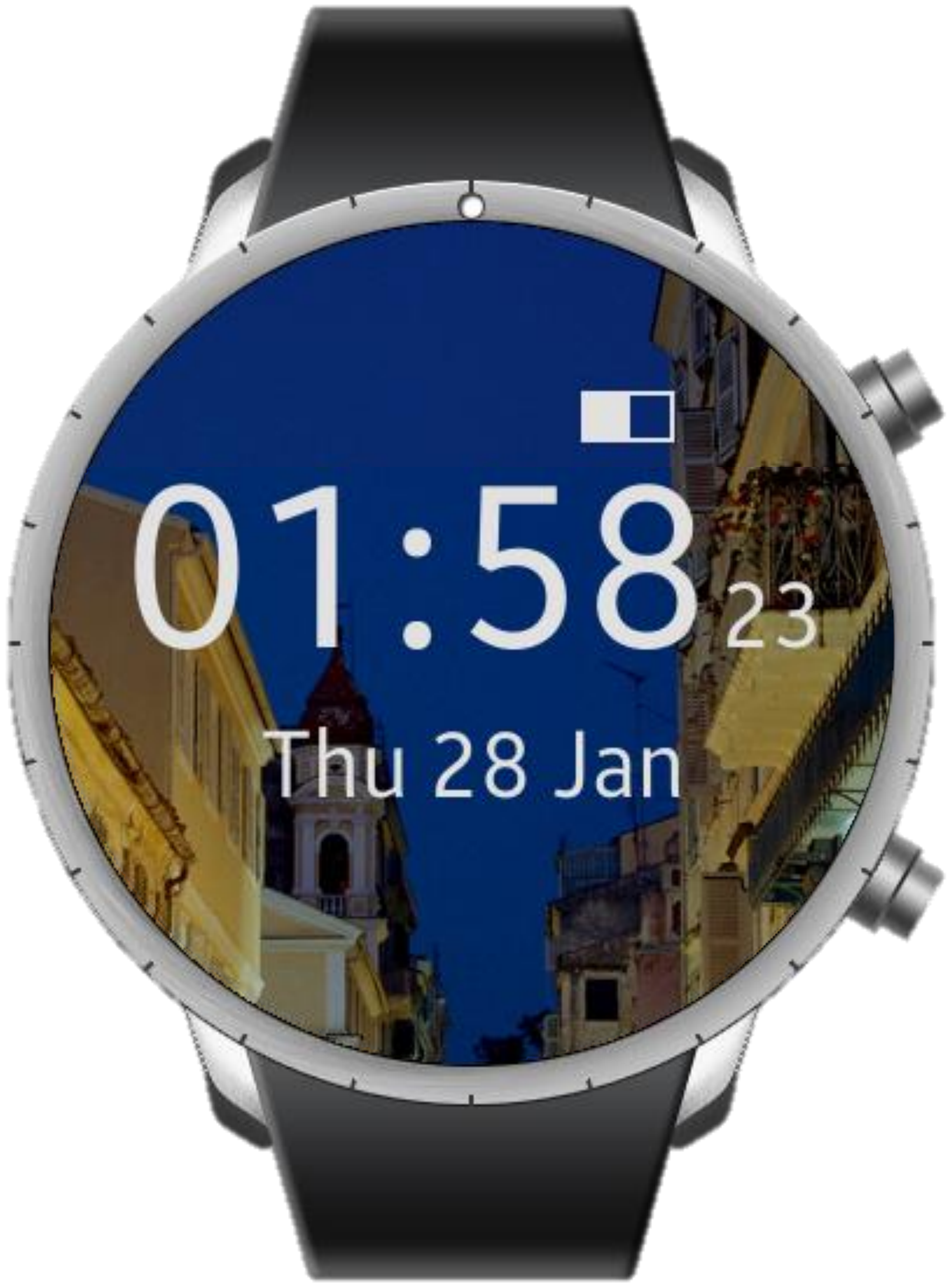
The following figure summarizes the interactions between the components:



## Implementing Tizen Web Applications

## Implementing Tizen Web Applications – Digital Watch

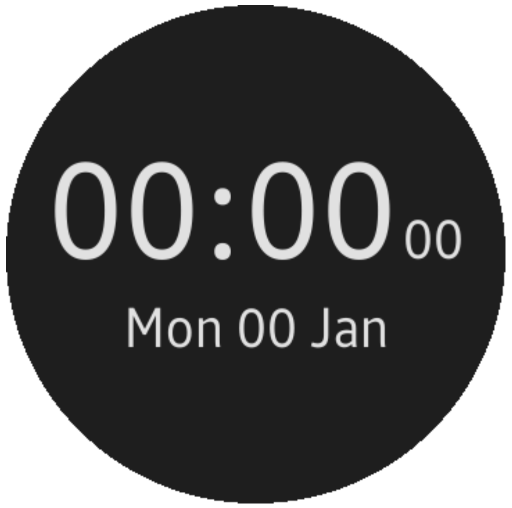
With the understanding of the basic components of Tizen Web applications, you can create a simple digital watch application.



## Implementation Plan

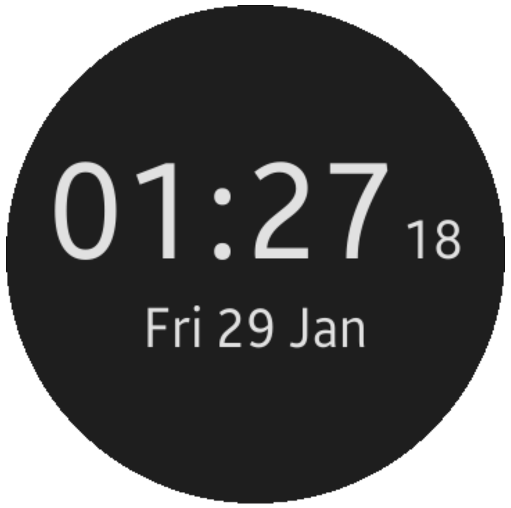
The digital watch application is implemented in 4 stages.

**Stage 1**
Digital watch user interface layout



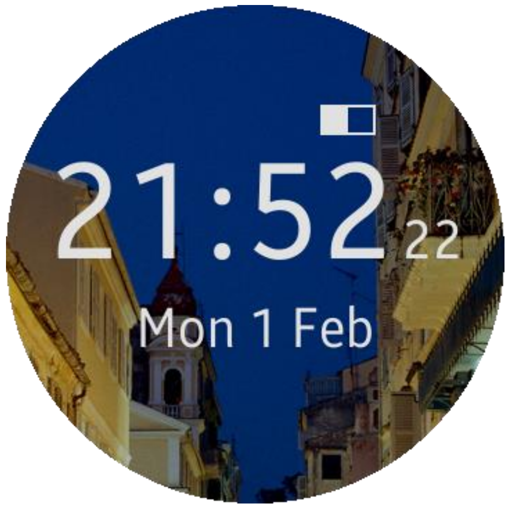
Texts

**Stage 2**
Digital watch functionality



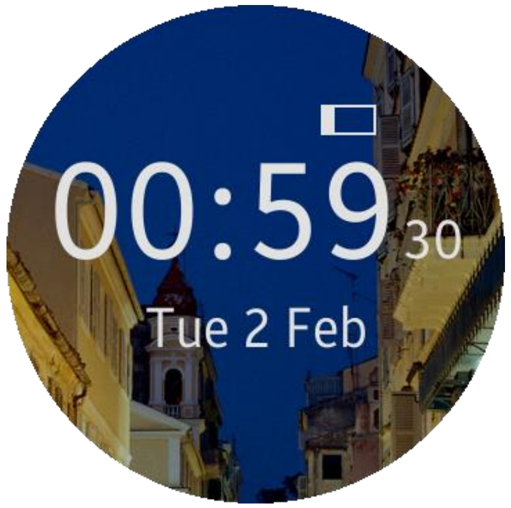
W3C Standard API

**Stage 3**
Battery user interface layout



Box and Background

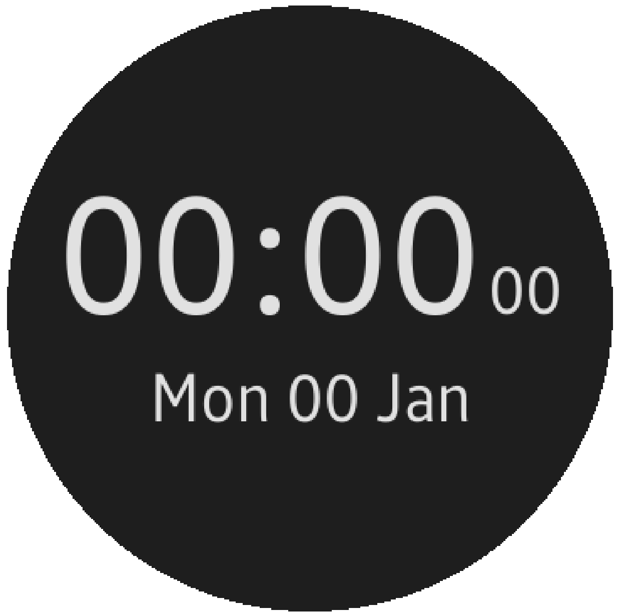
**Stage 4**
Battery functionality



Tizen Device API

## Stage 1: Digital Watch UI Layout – Goal

The goal of Stage 1 is to implement the user interface layout of a digital watch application using HTML and CSS.



In Stage 1-0, the codes for the implementation are preset.

In Stage 1-1, the layout of the digital watch application is defined using HTML `<div>` and `<span>` elements.

In Stage 1-2, the styling to the HTML elements is added using CSS properties, including `font-size` and `margin-top`.

In Stage 1-3, the stage is summarized.

## Stage 1-0: Digital Watch UI Layout – Pre-setting

Start the implementation by resetting the codes in the Basic template application (myFirstApp application).
Delete the unnecessary code in the `index.html` file of the myFirstApp application.

index.html

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8" />
    <meta name="viewport"
          content="width=device-width, initial-scale=1.0, maximum-scale=1.0">
    <meta name="description" content="Tizen Wearable Web Basic Template" />

    <title>Tizen Wearable Web Basic Application</title>

    <link rel="stylesheet" type="text/css" href="css/style.css" />
    <script src="js/main.js"></script>
</head>

<body>
    <div id="main" class="page">
        <div class="contents">
            <span id="content-text">Basic</span>
        </div>
    </div>
</body>
</html>
```

Delete the unnecessary code

## Stage 1-0: Digital Watch UI Layout – Pre-setting

Delete the unnecessary code in the `style.css` file of the myFirstApp application.

style.css

```
html,
body {
    width: 100%;
    height: 100%;
    margin: 0 auto;
    padding: 0;
    background-color: #222222;
    color: #ffffff;
}
.page {
    width: 100%;
    height: 100%;
    display: table;
}
.contents {
    display: table-cell;
    vertical-align: middle;
    text-align: center;
    -webkit-tap-highlight-color: transparent;
}
#content-text {
    font-weight: bold;
    font-size: 5em;
}
```

→ Delete the unnecessary code

# Stage 1-0: Digital Watch UI Layout – Pre-setting

Delete the unnecessary code in the `main.js` file of the myFirstApp application.

`main.js`

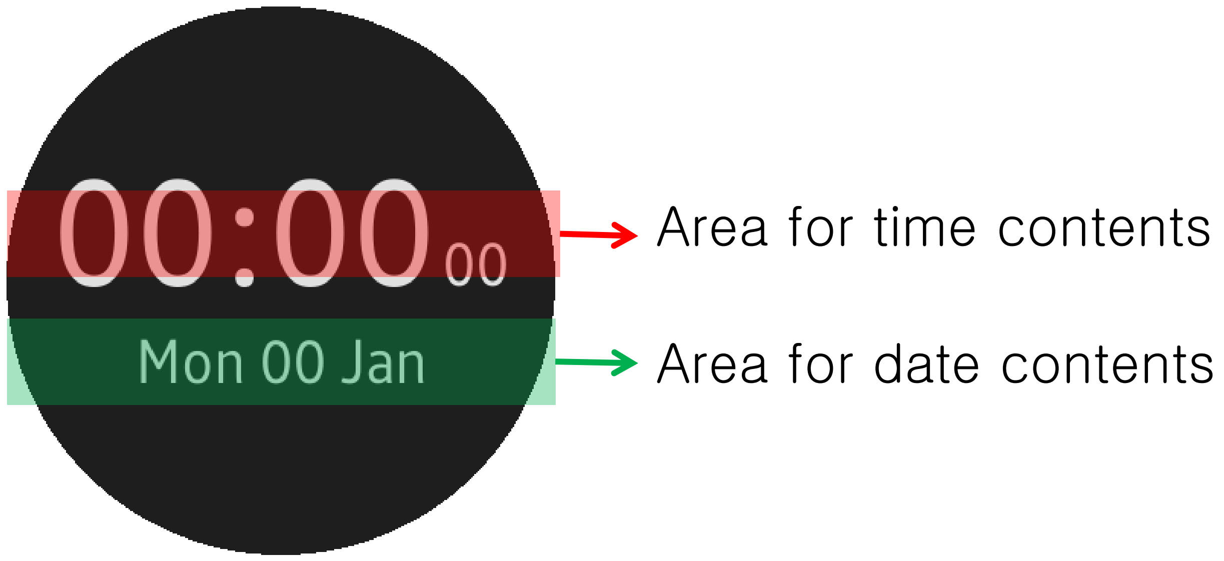
```
window.onload = function() {
    // TODO:: Do your initialization job

    // Add eventListener for tizenhwkey
    document.addEventListener('tizenhwkey', function(e) {
        if (e.keyName === "back") {
            try {
                tizen.application.getCurrentApplication().exit();
            } catch (ignore) {}
        }
    });

    // Sample code
    var mainPage = document.querySelector('#main');

    mainPage.addEventListener("click", function() {
        var contentText = document.querySelector('#content-text');

        contentText.innerHTML = (contentText.innerHTML === "Basic") ? "Tizen" : "Basic";
    });
};
```

Delete the unnecessary code

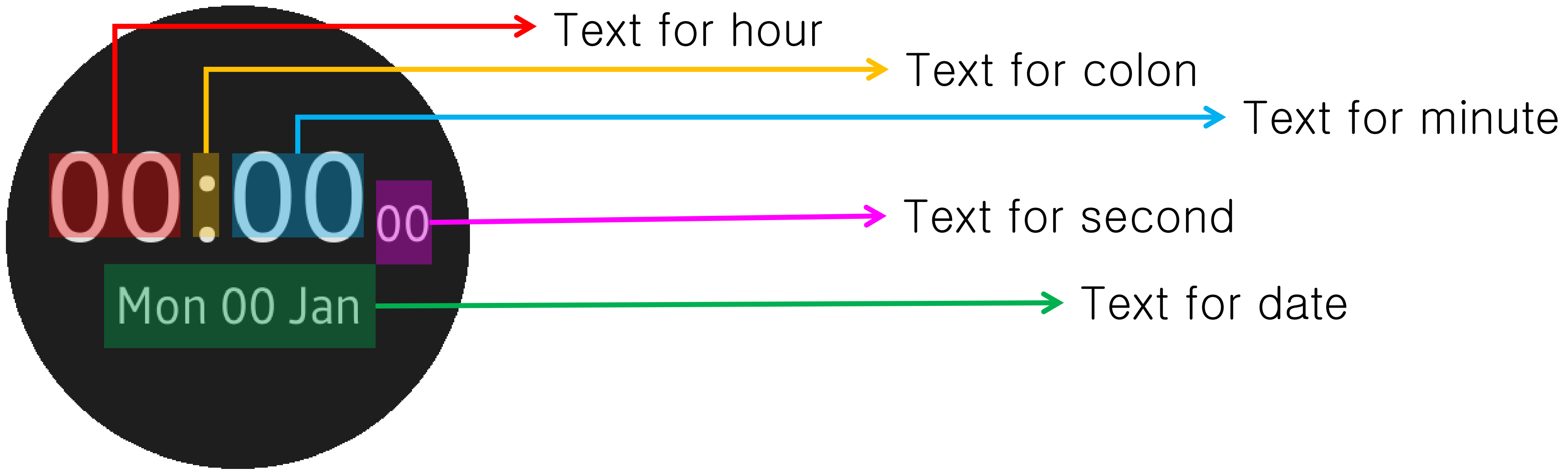
## Stage 1-1: Digital Watch UI Layout – index.html

Plan how to put each element in the `index.html` file.

You need an area for the time contents and another area for the date contents. Use `<div>` elements to create the areas.



Then, you need texts for the hours, the colon, the minutes, the seconds, and the date. Use `<span>` elements to create the texts.



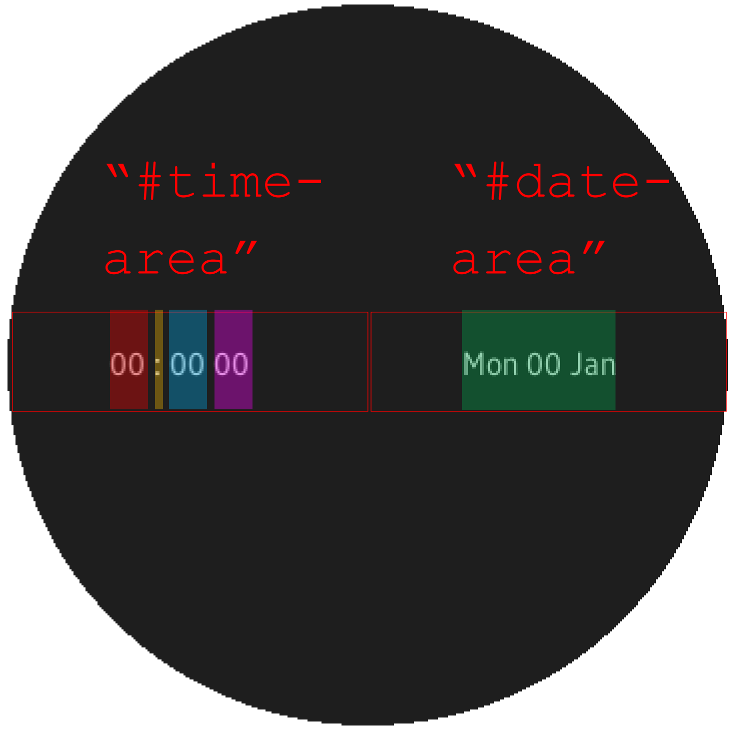
## Stage 1-1: Digital Watch UI Layout – index.html

Put all elements in the `index.html` file and fill in the contents with pseudo data.

`index.html`

```
...
<body>
    <div id="main" class="page">
        <div id="time-area" class="contents">
            <span id="hour-text">00</span>
            <span id="colon-text">:</span>
            <span id="minute-text">00</span>
            <span id="second-text">00</span>
        </div>
        <div id="date-area" class="contents">
            <span id="date-text">Mon 00 Jan</span>
        </div>
    </div>
</body>
...
```

Add new `<div>` and `<span>` elements



To learn more about the HTML `<div>` element, see:
[http://www.w3schools.com/tags/tag_div.asp](http://www.w3schools.com/tags/tag_div.asp)
To learn more about the HTML `<span>` element, see:
[http://www.w3schools.com/tags/tag_span.asp](http://www.w3schools.com/tags/tag_span.asp)

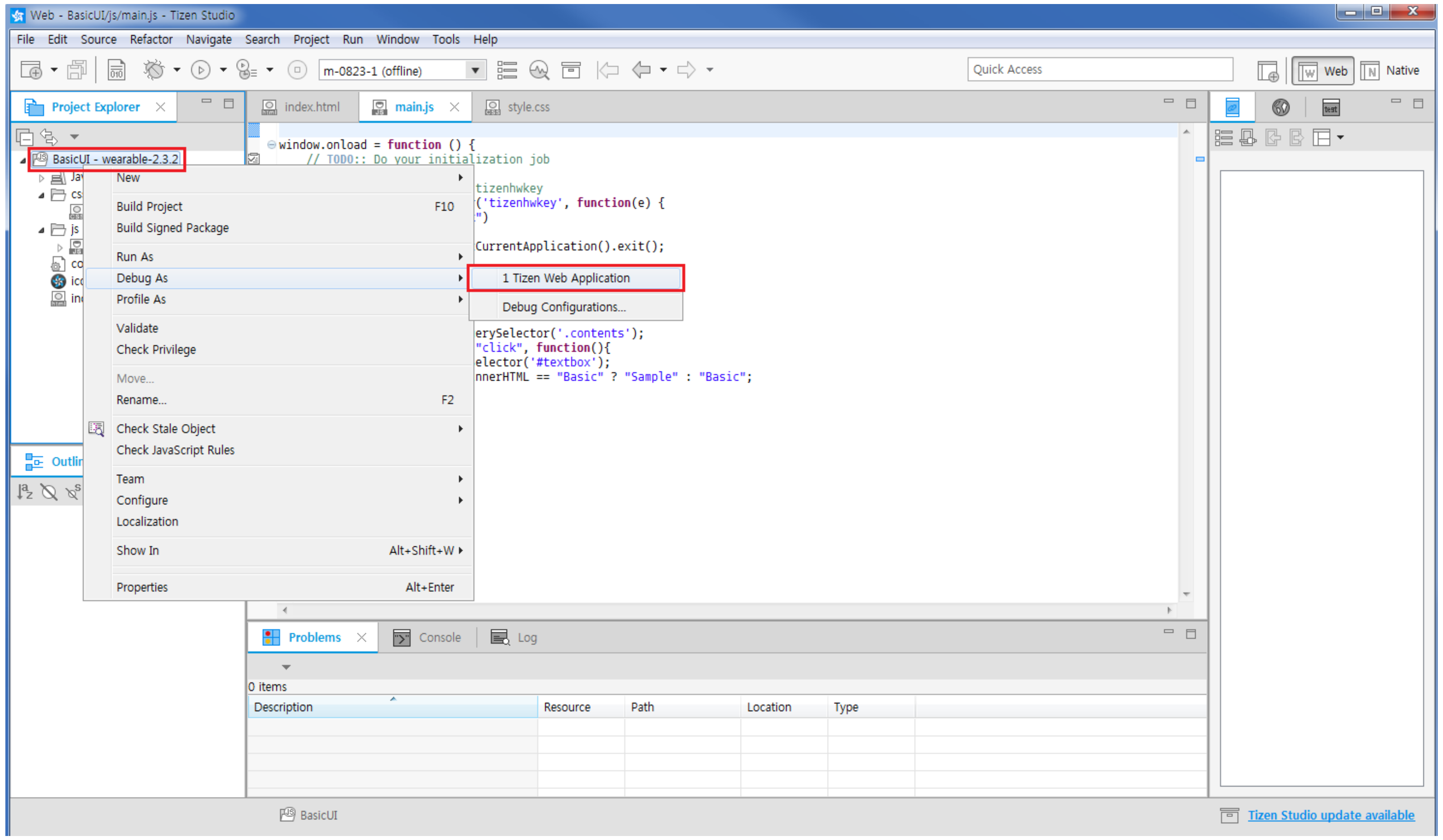
# Stage 1-2: Digital Watch UI Layout – style.css

In this stage, add styling to the HTML elements using the CSS properties.

To check how the elements are displayed right away, you can test the application in the **Web Inspector**, which is a debugging tool provided for Web applications.

To test the app in **Web Inspector**:

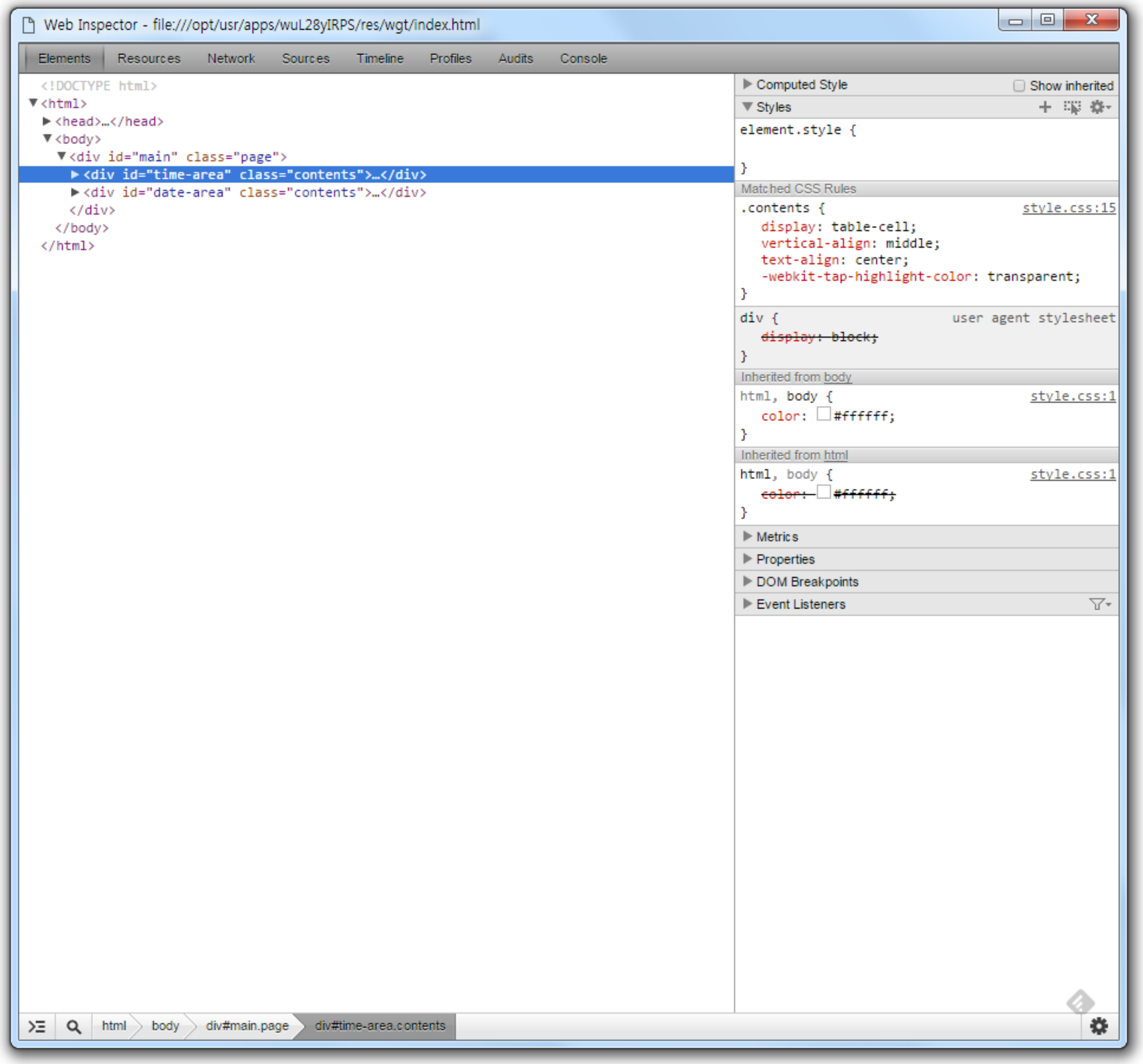
1. Right-click on the project name.
2. Select **Debug As > Tizen Web Application**.



```
window.onload = function () {
    // TODO:: Do your initialization job
tizenhwkey
('tizenhwkey', function(e) {
")
CurrentApplication().exit();
erySelector('.contents');
"click", function(){
elector('#textbox');
nnerHTML == "Basic" ? "Sample" : "Basic";
```

## Stage 1-2: Digital Watch UI Layout – Debugging

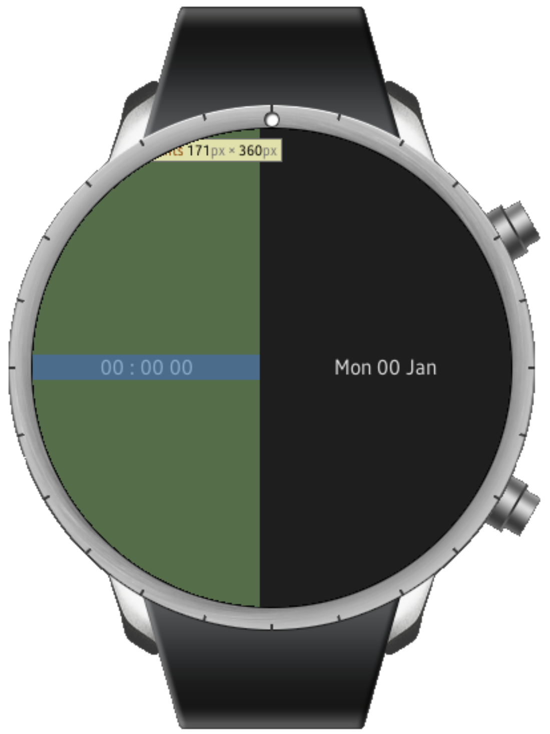
In the **Web Inspector**, select an element and its placement is shown in the emulator correspondingly.



Web Inspector - file:///opt/usr/apps/wuL28yIRPS/res/wgt/index.html

```
<!DOCTYPE html>
▼<html>
  ▶<head>…</head>
  ▼<body>
    ▼<div id="main" class="page">
      ▶<div id="time-area" class="contents">…</div>
      ▶<div id="date-area" class="contents">…</div>
     </div>
   </body>
</html>
```

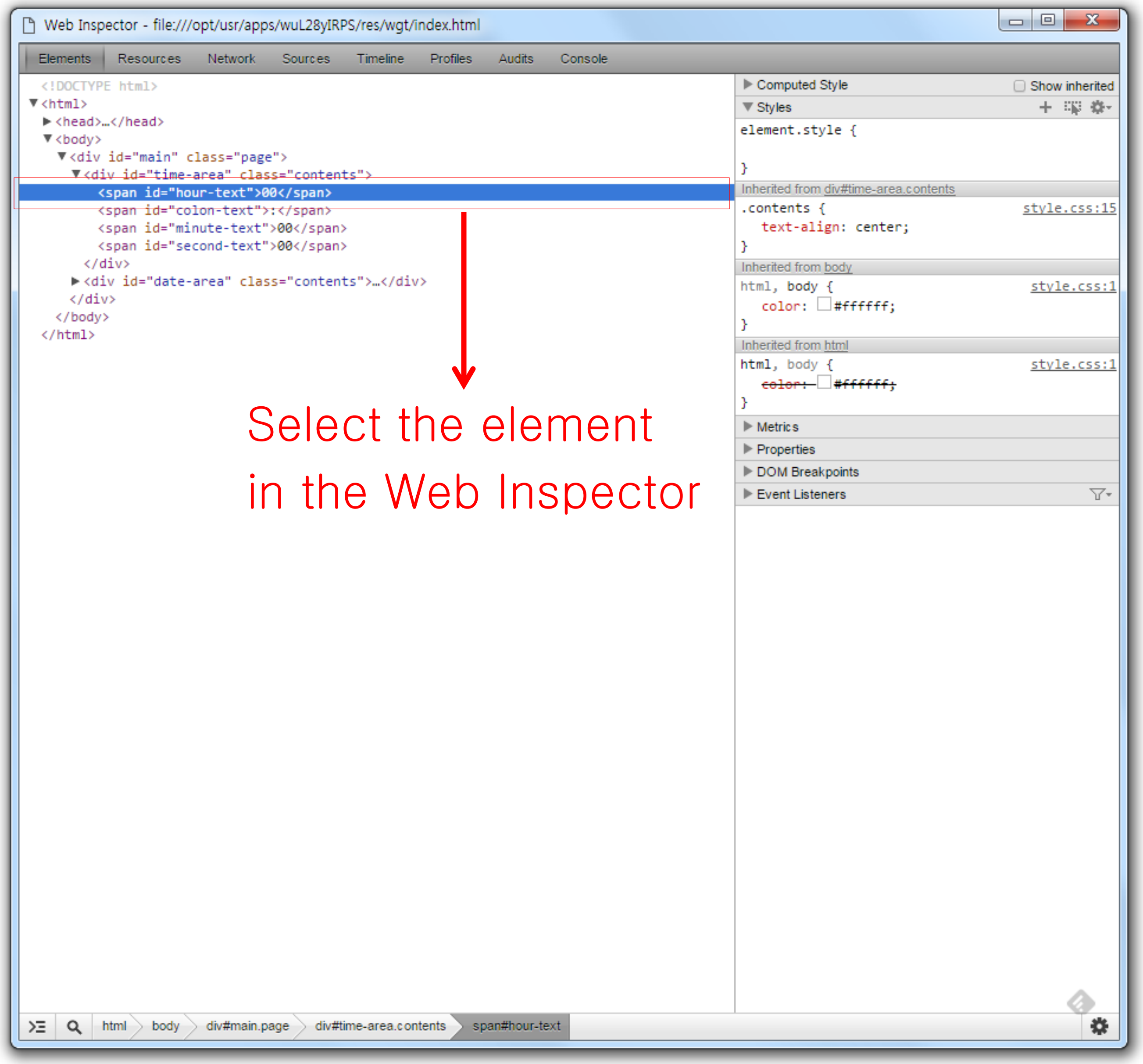
```
element.style {

}
Matched CSS Rules
.contents {                                    style.css:15
   display: table-cell;
   vertical-align: middle;
   text-align: center;
   -webkit-tap-highlight-color: transparent;
}
div {                              user agent stylesheet
   ~~display: block;~~
}
Inherited from body
html, body {                                    style.css:1
   color: #ffffff;
}
Inherited from html
html, body {                                    style.css:1
   ~~color: #ffffff;~~
}
```



## Stage 1-2: Digital Watch UI Layout – Debugging

Try adjusting the font size of `<span id="hour-text">00</span>`.

1. Select the element in the inspector.

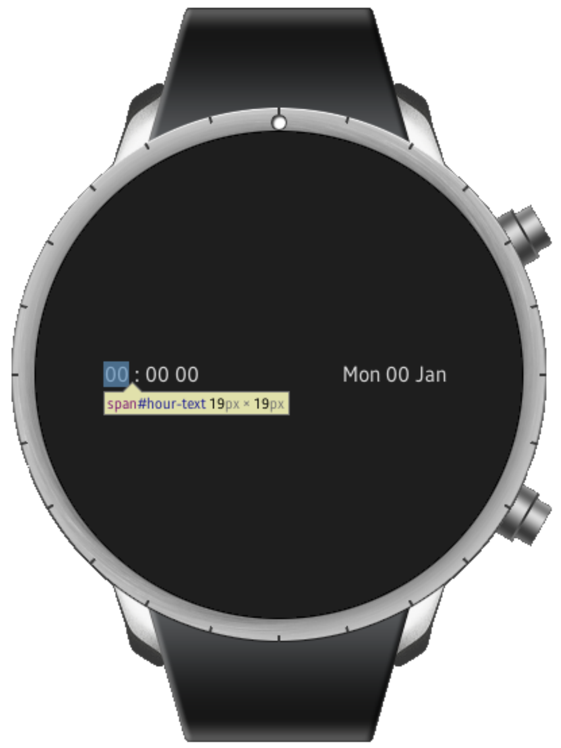


Web Inspector - file:///opt/usr/apps/wuL28yIRPS/res/wgt/index.html

```
<!DOCTYPE html>
▼<html>
  ▶<head>…</head>
  ▼<body>
    ▼<div id="main" class="page">
      ▼<div id="time-area" class="contents">
          <span id="hour-text">00</span>
          <span id="colon-text">:</span>
          <span id="minute-text">00</span>
          <span id="second-text">00</span>
        </div>
      ▶<div id="date-area" class="contents">…</div>
    </div>
  </body>
</html>
```

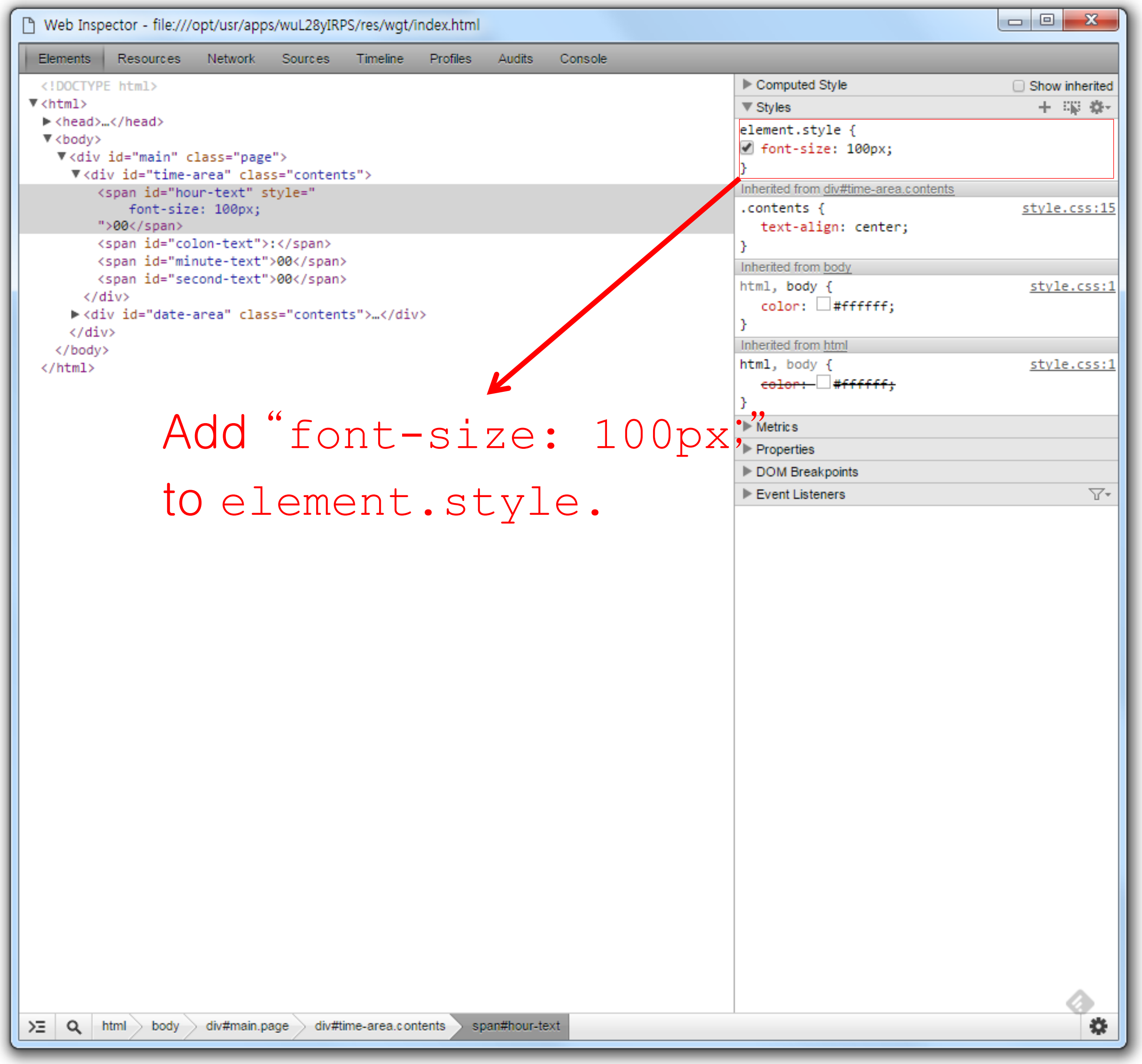
```
element.style {

}
Inherited from div#time-area.contents
.contents {                          style.css:15
   text-align: center;
}
Inherited from body
html, body {                         style.css:1
   color: #ffffff;
}
Inherited from html
html, body {                         style.css:1
   ~~color: #ffffff;~~
}
```

Select the element
in the Web Inspector



## Stage 1-2: Digital Watch UI Layout – Debugging

2. Add "`font-size: 100px;`" to `element.style` on the right column.

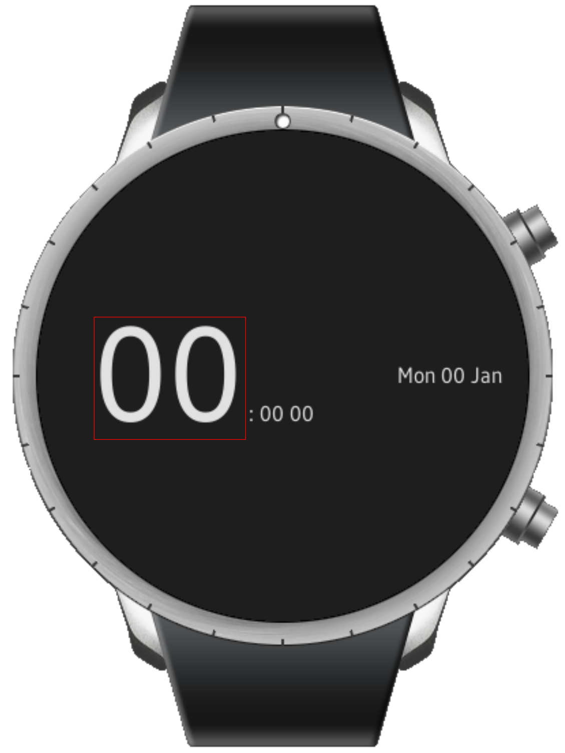


Web Inspector - file:///opt/usr/apps/wuL28yIRPS/res/wgt/index.html

```
<!DOCTYPE html>
▼<html>
  ▶<head>…</head>
  ▼<body>
    ▼<div id="main" class="page">
      ▼<div id="time-area" class="contents">
          <span id="hour-text" style="
              font-size: 100px;
          ">00</span>
          <span id="colon-text">:</span>
          <span id="minute-text">00</span>
          <span id="second-text">00</span>
        </div>
      ▶<div id="date-area" class="contents">…</div>
      </div>
    </body>
</html>
```

```
element.style {
✓ font-size: 100px;
}
Inherited from div#time-area.contents
.contents {                              style.css:15
   text-align: center;
}
Inherited from body
html, body {                             style.css:1
   color: #ffffff;
}
Inherited from html
html, body {                             style.css:1
   ~~color: #ffffff;~~
}
```

Add "`font-size: 100px;`" to `element.style`.



The size of the font is changed.

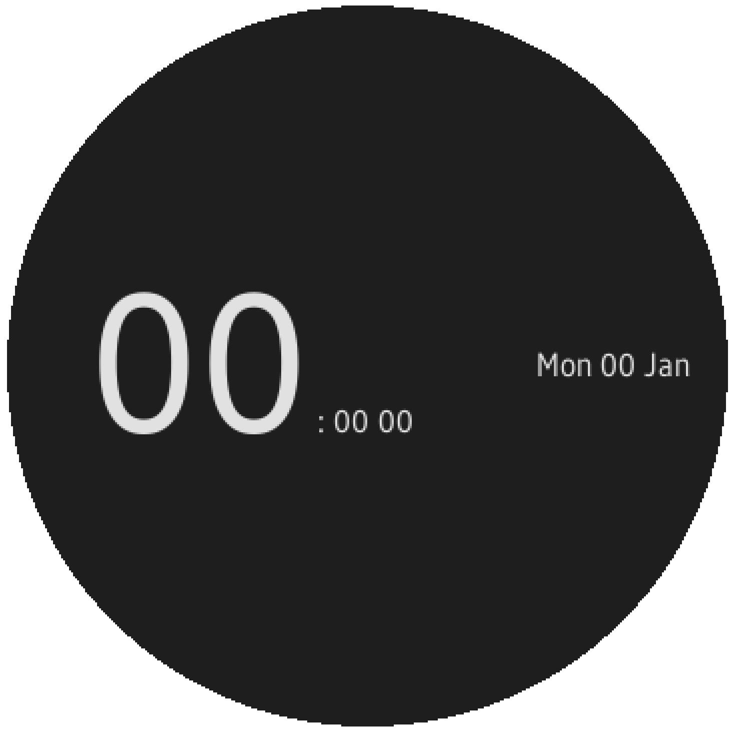
## Stage 1-2: Digital Watch UI Layout – Debugging

The next step is to actually apply the change in the `style.css` file.

style.css

```
...

.contents {
    ...
}
#hour-text {
    font-size: 100px;
}
```

Apply the change.



You can see that the `<div>` elements are aligned horizontally.
This is because the CSS display property of the `<div>` elements is set to "`table-cell`".

Next, change the code in the `style.css` file to align the elements vertically.

To learn more about the CSS `font-size` property, see:
http://www.w3schools.com/cssref/pr_font_font-size.asp

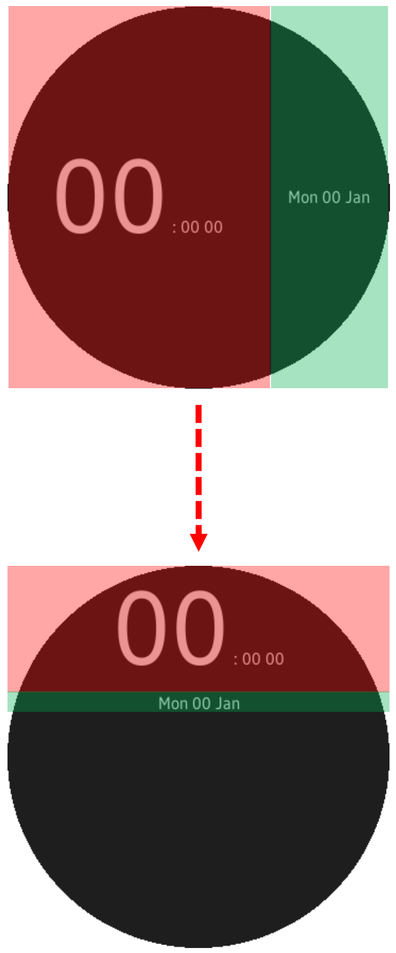
# Stage 1-2: Digital Watch UI Layout – style.css

Change the codes in the `style.css` file to change the display property of the `<div>` elements with a class name "`contents`".

`style.css`

```
/* old */
.contents {
    display: table-cell;
    text-align: center;
    vertical-align: middle;
    -webkit-tap-highlight-color: transparent;
}

/* new */
.contents {
    display: block;
    text-align: center;
    vertical-align: middle;
    -webkit-tap-highlight-color: transparent;
}
```

Change the value of "`.contents`" selector's "`display`" property from "`table-cell`" to "`block`".



The texts are no longer aligned vertically in the middle, because the `vertical-align` property does not work with the block display.

To learn more about the CSS `display` property, see:
[http://www.w3schools.com/css/css_display_visibility.asp](http://www.w3schools.com/css/css_display_visibility.asp)

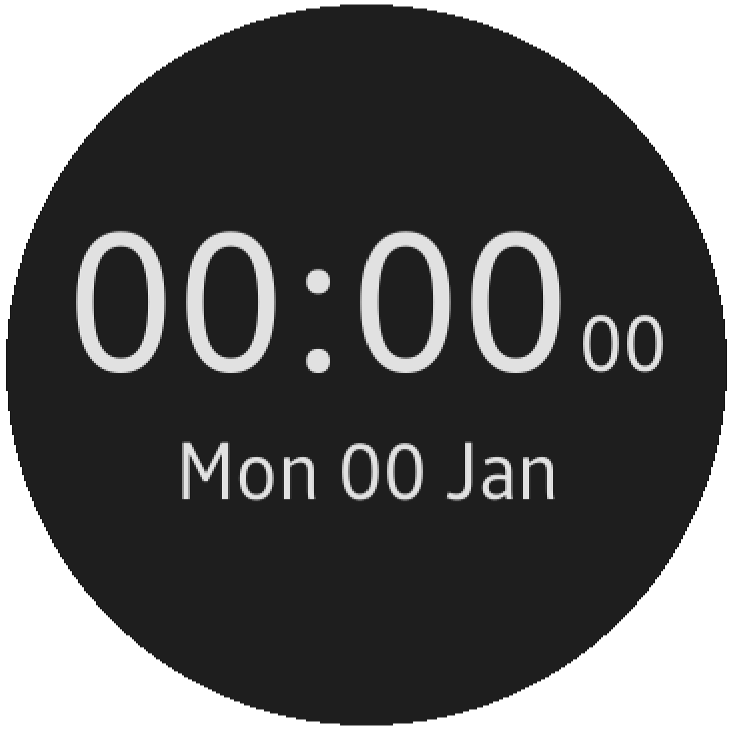
## Stage 1-2: Digital Watch UI Layout – style.css

Add all the necessary styling in the `style.css` file.

`style.css`

```
...

#hour-text {
    ...
}
#time-area {
    margin-top: 25%;
}

#second-text, #date-text {
    font-size: 40px;
}

#colon-text, #minute-text {
    font-size: 100px;
}
```

Add styling to the texts

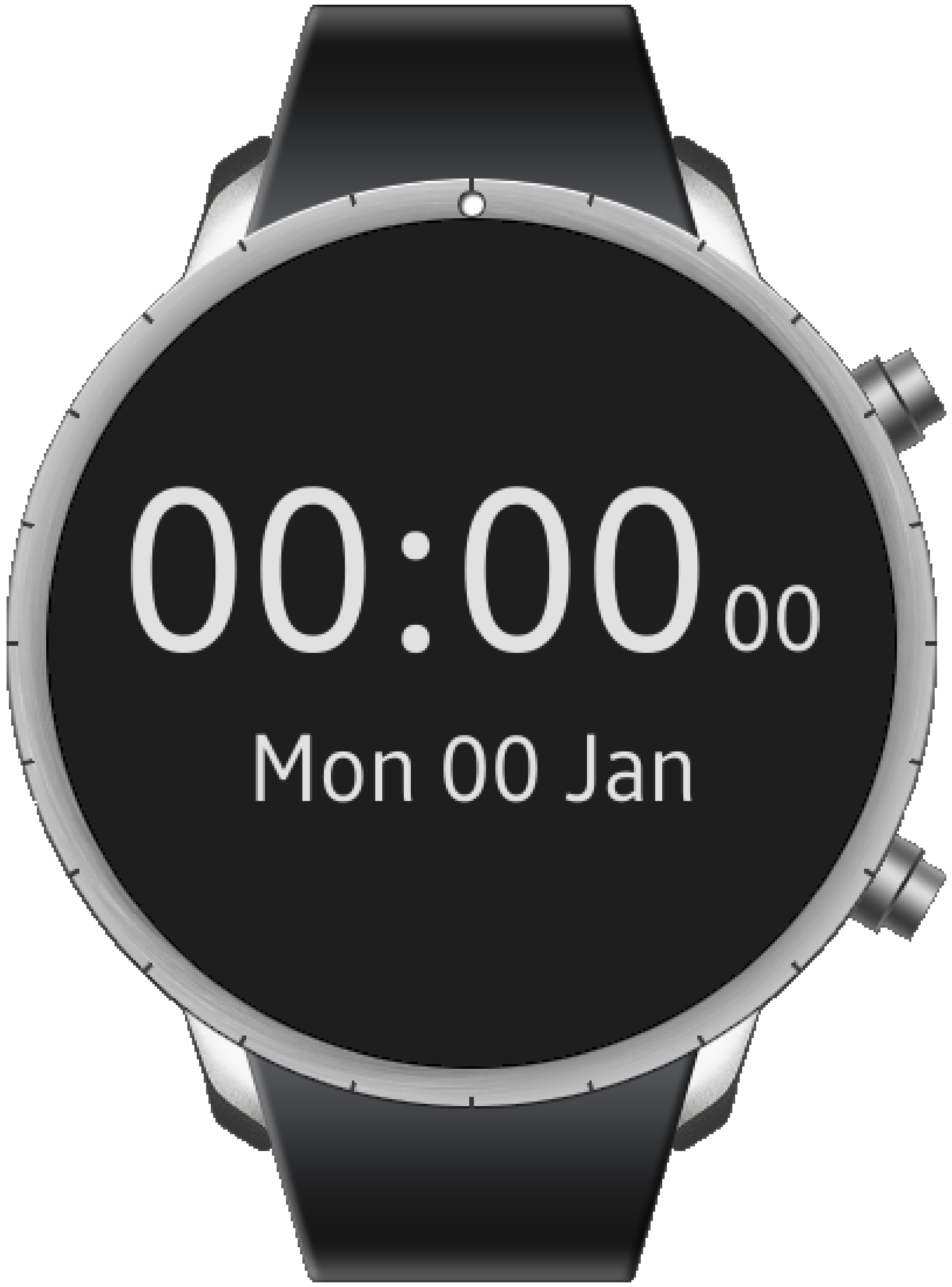


The Stage 1 is completed, and the layout has been created.

To learn more about the CSS `margin` property, see:
http://www.w3schools.com/css/css_margin.asp

## Stage 1-3: Digital Watch UI Layout – Summary

By the end of Stage 1, your application should appear as in the following figure:



# Stage 1-3: Digital Watch UI Layout – Summary

Your `index.html` file should appear as in the following example:

index.html

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8" />
    <meta name="viewport"
          content="width=device-width, initial-scale=1.0, maximum-scale=1.0">
    <meta name="description" content="Tizen Wearable Web Basic Template" />
    <title>Tizen Wearable Web Basic Application</title>

    <link rel="stylesheet" type="text/css" href="css/style.css" />
    <script src="js/main.js"></script>
</head>

<body>
    <div id="main" class="page">
        <div id="time-area" class="contents">
            <span id="hour-text">00</span>
            <span id="colon-text">:</span>
            <span id="minute-text">00</span>
            <span id="second-text">00</span>
        </div>
        <div id="date-area" class="contents">
            <span id="date-text">Mon 00 Jan</span>
        </div>

    </div>
</body>
</html>
```

## Stage 1-3: Digital Watch UI Layout – Summary

Your `style.css` file should appear as in the following example:

style.css

```
html,
body {
    width: 100%;
    height: 100%;
    margin: 0 auto;
    padding: 0;
    background-color: #222222;
    color: #ffffff;
}
.page {
    width: 100%;
    height: 100%;
    display: table;
}
```

The code continues on the right

```
.contents {
    display: block;
    vertical-align: middle;
    text-align: center;
    -webkit-tap-highlight-color:
transparent;
}
#hour-text {
    font-size: 100px;
}
#time-area {
    margin-top: 25%;
}
#second-text, #date-text {
    font-size: 40px;
}
#colon-text, #minute-text {
    font-size: 100px;
}
```

## Stage 1-3: Digital Watch UI Layout – Summary

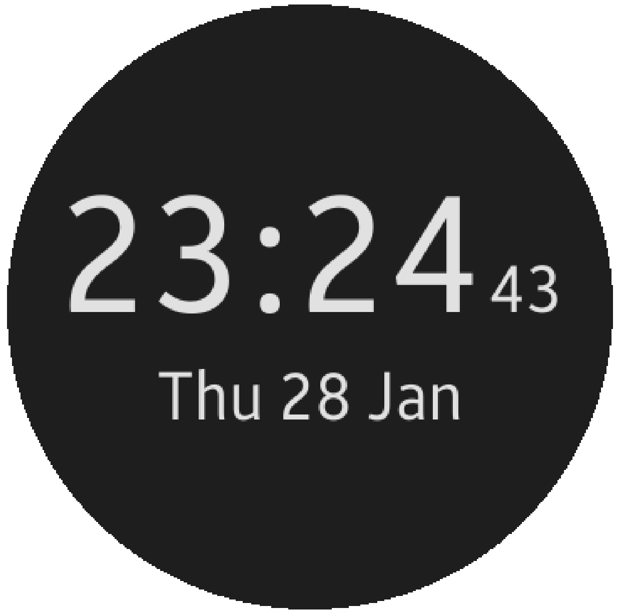
Lastly, your `main.js` file should appear as in the following example:

main.js

```
window.onload = function() {
    // TODO:: Do your initialization job

    // Add eventListener for tizenhwkey
    document.addEventListener('tizenhwkey', function(e) {
        if (e.keyName === "back") {
            try {
                tizen.application.getCurrentApplication().exit();
            } catch (ignore) {}
        }
    });
};
```

## Stage 2: Digital Watch Functionality – Goal

The goal of Stage 2 is to implement the functionality of the digital watch using JavaScript and W3C standard Web API.



In Stage 2-1, create a `changeHTML()` method using the `document.getElementById()` method and `element.innerHTML`.

In Stage 2-2, create a `changeTime()` method using the W3C `Date()` API.

In Stage 2-3, create a `changeDate()` method using the W3C `Date()` API.

In Stage 2-4, the stage is summarized.

## Stage 2-1: Digital Watch Functionality – changeHTML()

Take a look at the initial JavaScript codes in the `main.js` file.
The code in the `window.onload` method is executed when the window is loaded, which is when the application is launched.

`main.js`

```
window.onload = function() {
    // TODO:: Do your initialization job

    // Add eventListener for tizenhwkey
    document.addEventListener('tizenhwkey', function(e) {
        if (e.keyName === "back") {
            try {
                tizen.application.getCurrentApplication().exit();
            } catch (ignore) {}
        }
    });
};
```

The file contains a code block which closes the application when the back key is pressed, but do not worry about the code details for now.

To learn more about the JavaScript `onload` event, see:
[http://www.w3schools.com/jsref/event_onload.asp](http://www.w3schools.com/jsref/event_onload.asp)

## Stage 2-1: Digital Watch Functionality – changeHTML()

Remember that you can change the HTML contents using JavaScript?

Implement a simple method which grabs an HTML element and changes its content. The new method is named `changeHTML()`.
The method is added above the `window.onload` method.

`main.js`

```
function changeHTML(elemId, newContent) {
    var htmlElem = document.getElementById(elemId);

    htmlElem.innerHTML = newContent;
}

window.onload = function() {
    ...
};
```

Grabs a HTML element with the id '`elemId`' and stores the element object in the variable '`htmlElem`'

Changes the content of the '`htmlElem`' element with '`newContent`'

To learn more about the `document.getElementById()` method:
http://www.w3schools.com/jsref/met_document_getelementbyid.asp

To learn more about `element.innerHTML`:
http://www.w3schools.com/jsref/prop_html_innerhtml.asp

## Stage 2-2: Digital Watch Functionality – changeTime()

Now, change the contents with real data.
Create a new `changeTime()` method for changing the texts for the current hour and minute using the W3C Standard `Date()` API.

main.js

```
function changeHTML(elemId, newContent) {
    ...
}

function changeTime() {
    var date = new Date(),
        currentHour = date.getHours(),
        currentMinute = date.getMinutes(),
        currentSecond = date.getSeconds();

    changeHTML("hour-text", currentHour);
    changeHTML("minute-text",currentMinute);
    changeHTML("second-text",currentSecond);
}

window.onload = function() {
    ...
};
```

Get the values of current hour, minute, and second, and store the value in the variables.

Call the `changeHTML` method with the variables.

To learn more about the W3C `Date()` API:
http://www.w3schools.com/js/js_date_methods.asp

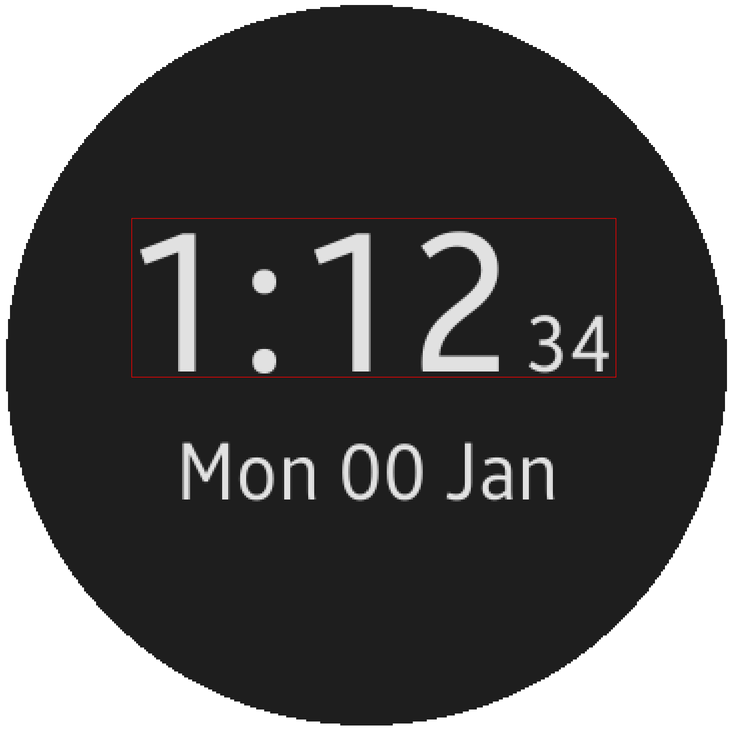
## Stage 2-2: Digital Watch Functionality – changeTime()

Then, call the `changeTime()` method in the `window.onload` method so that it gets executed when the application is launched.

Moreover, call the method inside the `setInterval()` method so that the method gets executed every 1 second as the time changes.

main.js

```
...

window.onload = function() {
    // TODO:: Do your initialization job
     setInterval(function() {
        changeTime();
    }, 1000);
    ...
};
```



To learn more about the `setInterval()` method, see:
http://www.w3schools.com/jsref/met_win_setinterval.asp

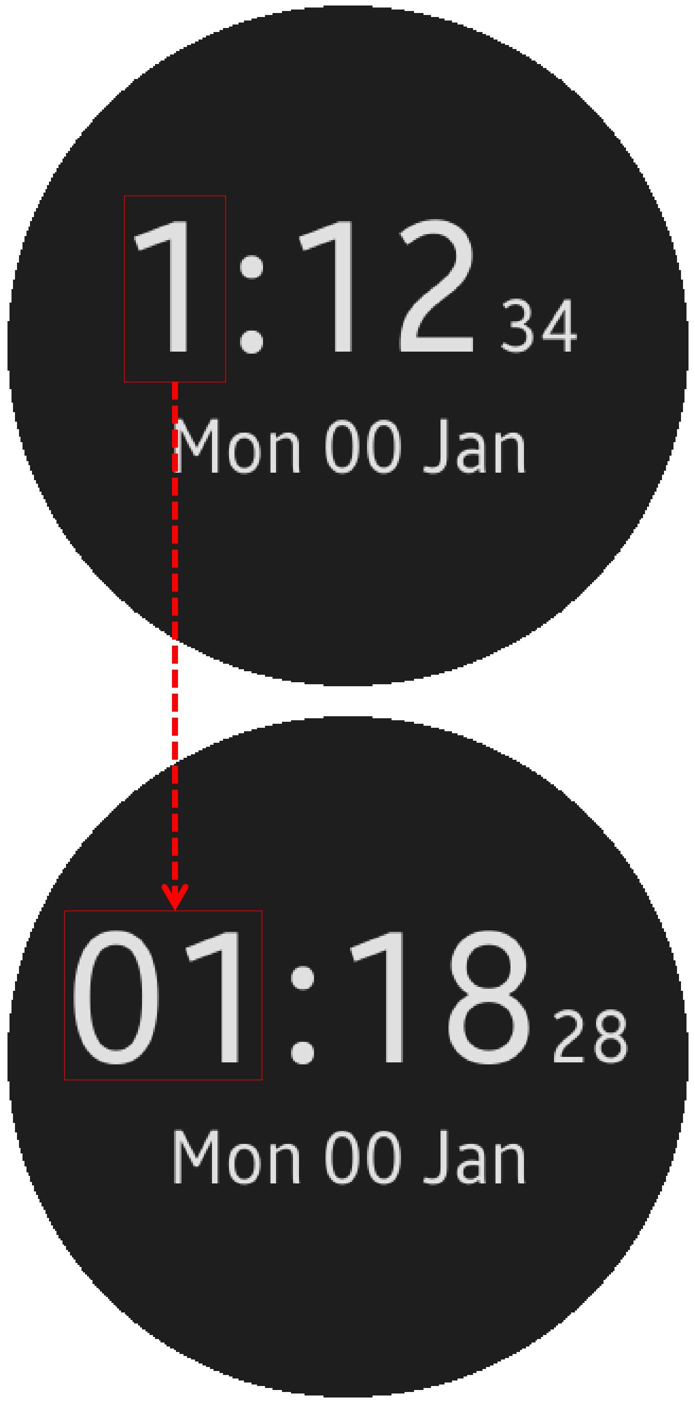
## Stage 2-2: Digital Watch Functionality – changeTime()

Modify the `changeTime()` method so that the hour text, the minute text, and the second text are always shown as double-digit numbers.

main.js

```
function changeTime() {
    ...

    // Make the hour, minute, and second texts show as
    // double-digit numbers
    if (currentHour < 10) {
        changeHTML("hour-text", "0" + currentHour);
    } else {
        changeHTML("hour-text", currentHour);
    }

    if (currentMinute < 10) {
        changeHTML("minute-text","0" + currentMinute);
    } else {
        changeHTML("minute-text",currentMinute);
    }

    if (currentSecond < 10) {
        changeHTML("second-text","0" + currentSecond);
    } else {
        changeHTML("second-text",currentSecond);
    }

    changeHTML("hour-text", currentHour();
    changeHTML("minute-text",currentMinute();
    changeHTML("second-text",currentSecond();
}
```

Delete the previous code.



## Stage 2-3: Digital Watch Functionality – changeDate()

The next step is to create a new `changeDate()` method for changing the texts for the current month, date, and day using the W3C Standard `Date()` API.

main.js

```
...

function changeTime() {
    ...
}

function changeDate() {
    var date = new Date(),
        currentMonth = date.getMonth(),
        currentDate = date.getDate(),
        currentDay = date.getDay(),
        strDate ="";

    strDate = currentDay + " " + currentDate + " " + currentMonth;

    changeHTML("date-text", strDate);
}

...
```

Get the values of current month, date and day and store the value in the variables.

Combine the values of day, date and month together to make a string in "Mon 00 Jan" format and store the value in "`strDate`" variable.

Call the `changeHTML` method with the "`strDate`" variable.

To learn more about the W3C `Date()` API, see:
http://www.w3schools.com/js/js_date_methods.asp

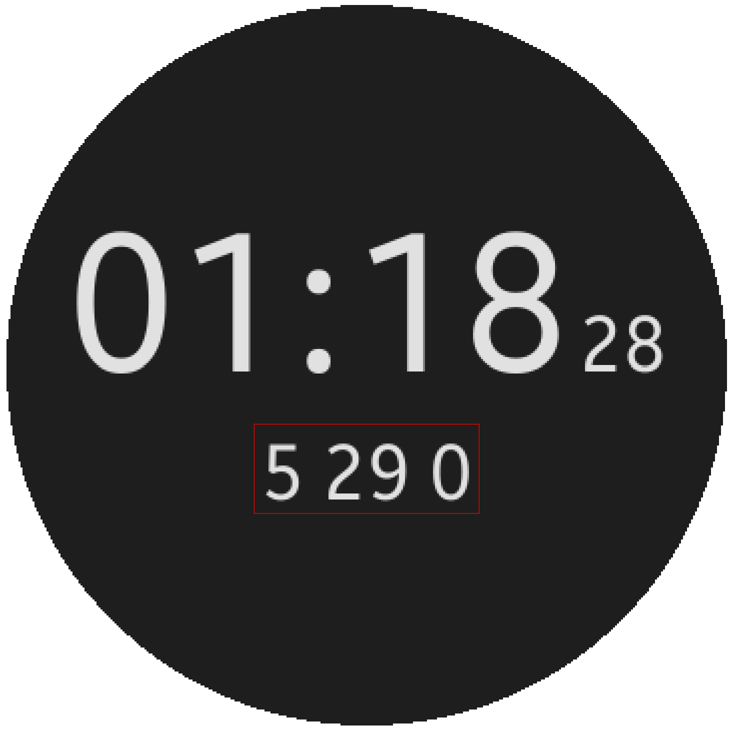
## Stage 2-3: Digital Watch Functionality – changeDate()

Call the `changeDate()` method in the `window.onload` method so that it gets executed when the application is launched.

`main.js`

```
...

window.onload = function() {
    // TODO:: Do your initialization job
    setInterval(function() {
        changeTime();
    }, 1000);

    changeDate();
    ...
};
```

Call the `changeDate()` method when the window is loaded



The text for the date does not appear as we expected because the W3C `Date()` API returns data for the month and day as an integer value.

Therefore, additional code is required for converting the integer values to the corresponding string values, such as for month data, converting 0 to "Jan".

## Stage 2-3: Digital Watch Functionality – changeDate()

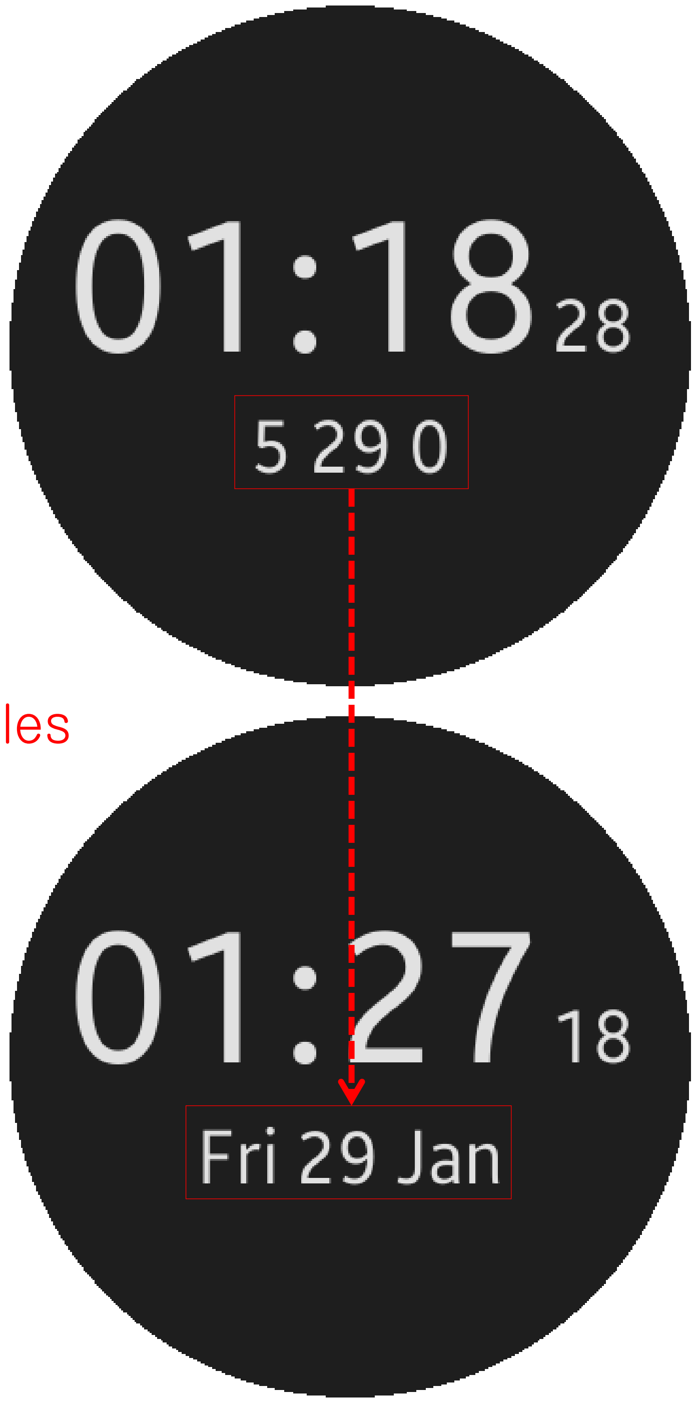
Modify the `changeDate()` method so that the integer values for month and day are converted to the corresponding string values.

main.js

```
function changeDate() {
    var date = new Date(),
        currentMonth = date.getMonth(),
        currentDate = date.getDate(),
        currentDay = date.getDay(),
        strDate ="",
        arrayMonth = ["Jan", "Feb", "Mar", "Apr", "May", "Jun",
                      "Jul", "Aug", "Sep", "Oct", "Nov", "Dec"],
        arrayDay = ["Sun", "Mon", "Tue", "Wed",
                    "Thu", "Fri", "Sat"];

    currentMonth = arrayMonth[currentMonth];
    currentDay = arrayDay[currentDay];

    strDate = currentDay + " " + currentDate + " " +
currentMonth;

    changeHTML("date-text", strDate);
}
```

Add arrays for string values of month and day

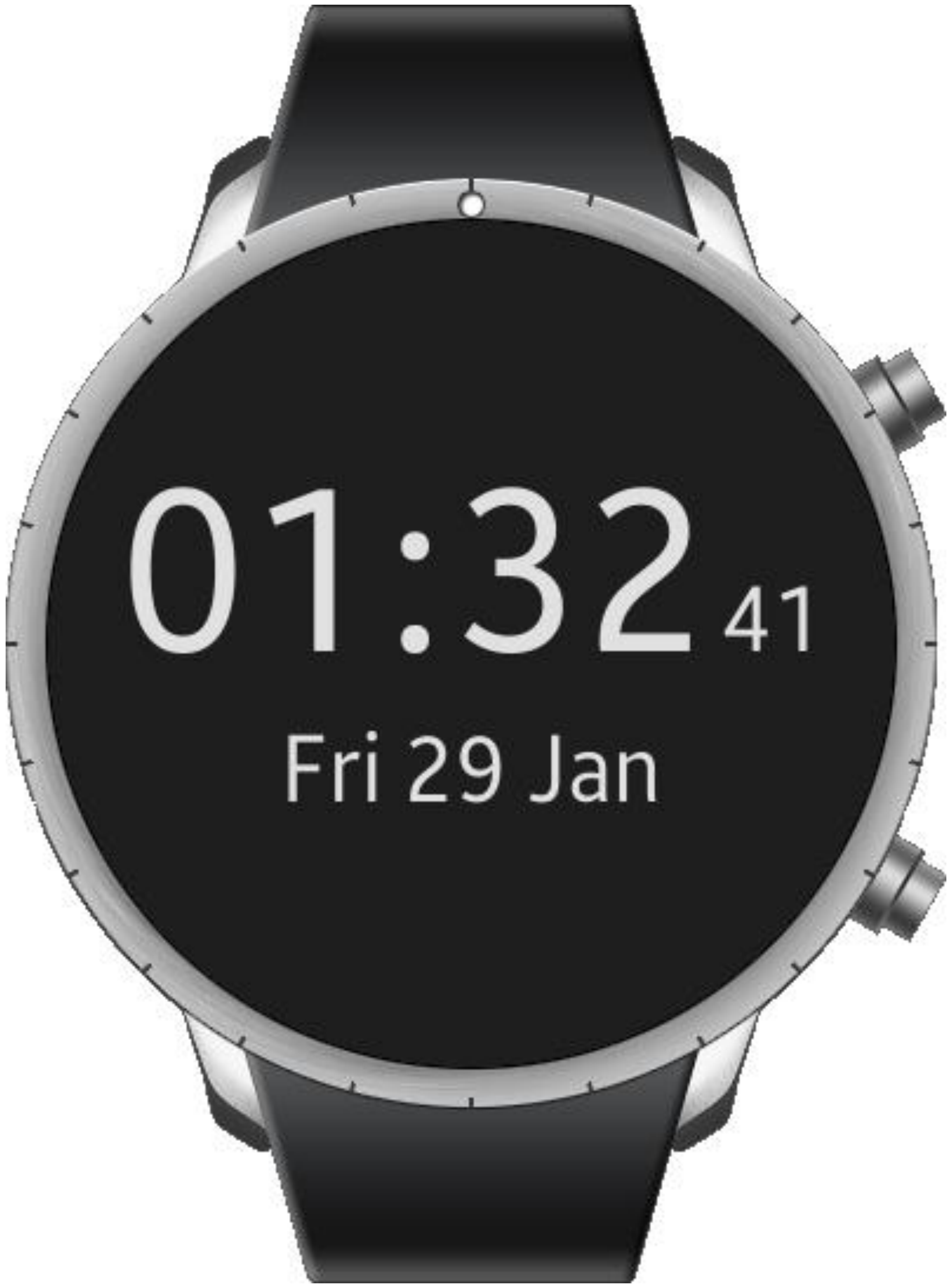
Reset the variables to string values



The Stage 2 is now completed, and the functionality is implemented.

## Stage 2-4: Digital Watch Functionality - Summary

By the end of Stage 2, your application should appear as in the following figure:



## Stage 2–4: Digital Watch Functionality – Summary

Your `index.html` file should appear as in the following example:

`index.html`

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0, maximum-
scale=1.0">
    <meta name="description" content="Tizen Wearable Web Basic Template" />
    <title>Tizen Wearable Web Basic Application</title>

    <link rel="stylesheet" type="text/css" href="css/style.css" />
    <script src="js/main.js"></script>
</head>

<body>
    <div id="main" class="page">
        <div id="time-area" class="contents">
            <span id="hour-text">00</span>
            <span id="colon-text">:</span>
            <span id="minute-text">00</span>
            <span id="second-text">00</span>
        </div>
        <div id="date-area" class="contents">
            <span id="date-text">Mon 00 Jan</span>
        </div>

    </div>
</body>
</html>
```

## Stage 2-4: Digital Watch Functionality – Summary

Your `style.css` file should appear as in the following example:

style.css

```
html,
body {
    width: 100%;
    height: 100%;
    margin: 0 auto;
    padding: 0;
    background-color: #222222;
    color: #ffffff;
}
.page {
    width: 100%;
    height: 100%;
    display: table;
}
```

The code continues on the right.

```
.contents {
    display: block;
    vertical-align: middle;
    text-align: center;
    -webkit-tap-highlight-color:
transparent;
}
#hour-text {
    font-size: 100px;
}
#time-area {
    margin-top: 25%;
}
#second-text, #date-text {
    font-size: 40px;
}
#colon-text, #minute-text {
    font-size: 100px;
}
```

## Stage 2–4: Digital Watch Functionality – Summary

Lastly, your `main.js` file should appear as in the following example:

main.js

```
function changeHTML(elemId, newContent) {
    var htmlElem = document.getElementById(elemId);

    htmlElem.innerHTML = newContent;
}
```

The code continues on the next page.

## Stage 2-4: Digital Watch Functionality – Summary

main.js

```javascript
function changeTime() {
    var date = new Date(),
        currentHour = date.getHours(),
        currentMinute = date.getMinutes(),
        currentSecond = date.getSeconds();

    // Make the hour, minute, and second texts show as double-digit numbers
    if (currentHour < 10) {
        changeHTML("hour-text", "0" + currentHour);
    } else {
        changeHTML("hour-text", currentHour);
    }

    if (currentMinute < 10) {
        changeHTML("minute-text", "0" + currentMinute);
    } else {
        changeHTML("minute-text", currentMinute);
    }

    if (currentSecond < 10) {
        changeHTML("second-text", "0" + currentSecond);
    } else {
        changeHTML("second-text", currentSecond);
    }
}
```

The code continues on the next page.

## Stage 2-4: Digital Watch Functionality – Summary

main.js

```
function changeDate() {
    var date = new Date(),
        currentMonth = date.getMonth(),
        currentDate = date.getDate(),
        currentDay = date.getDay(),
        strDate ="",
        arrayMonth = ["Jan", "Feb", "Mar", "Apr", "May", "Jun",
                      "Jul", "Aug", "Sep", "Oct", "Nov", "Dec" ],
        arrayDay = ["Sun", "Mon", "Tue", "Wed", "Thu", "Fri", "Sat"];

    currentMonth = arrayMonth[currentMonth];
    currentDay = arrayDay[currentDay];

    strDate = currentDay + " " + currentDate + " " + currentMonth;

    changeHTML("date-text", strDate);
}
```

The code continues on the next page.

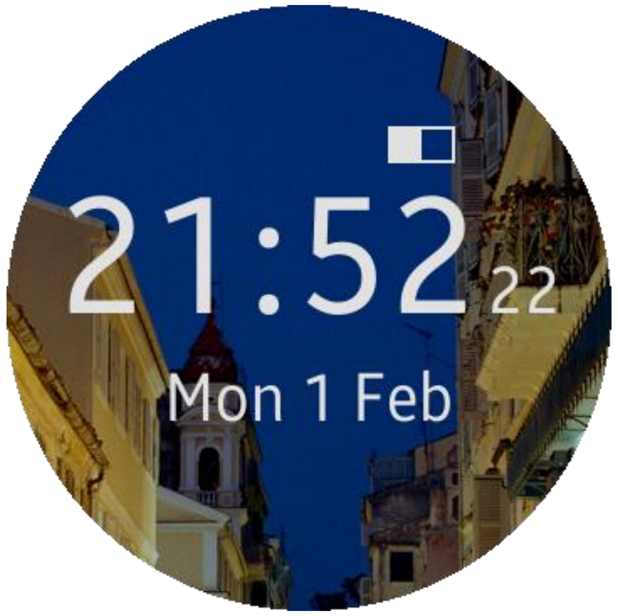
## Stage 2-4: Digital Watch Functionality – Summary

main.js

```
window.onload = function() {
    // TODO:: Do your initialization job
    setInterval(function() {
        changeTime();
    }, 1000);

    changeDate();

    // Add eventListener for tizenhwkey
    document.addEventListener('tizenhwkey', function(e) {
        if (e.keyName === "back") {
            try {
                tizen.application.getCurrentApplication().exit();
            } catch (ignore) {}
        }
    });
};
```

## Stage 3: Battery UI Layout – Goal

The goal of Stage 3 is to implement the user interface layout for the battery, and add a background image to the digital watch application using HTML and CSS.



In Stage 3-1, define the layout of the digital watch application using the HTML `<div>` elements.

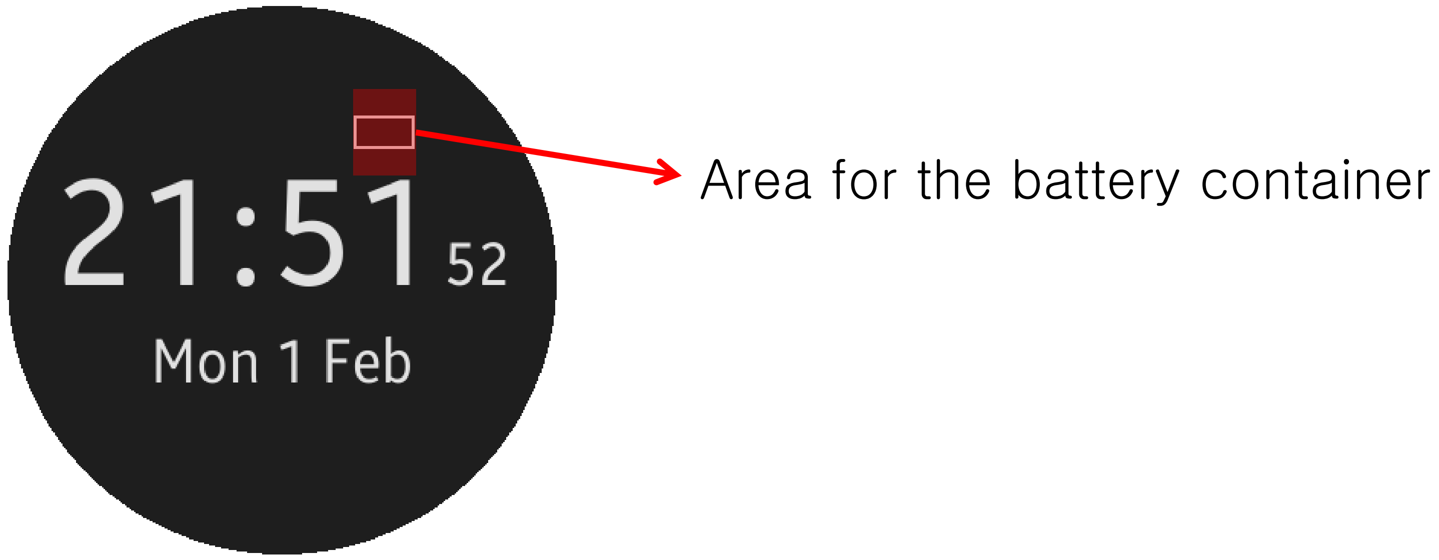
In Stage 3-2, add styling to the HTML elements using CSS properties, including a border and a background.

In Stage 3-3, the stage is summarized.

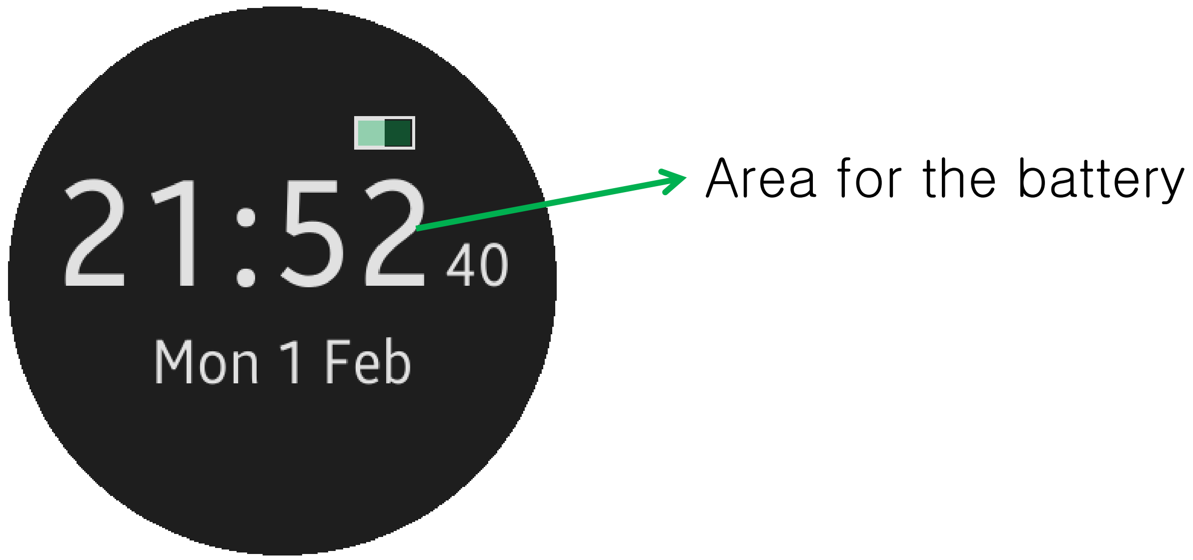
## Stage 3-1: Battery UI Layout – index.html

Plan how to place each element in the `index.html` file .

You need to create an area using `<div>` elements for the container of the battery element.



You need to create another area using `<div>` elements inside the container area for the actual battery element, which is filled accordingly with the battery level.

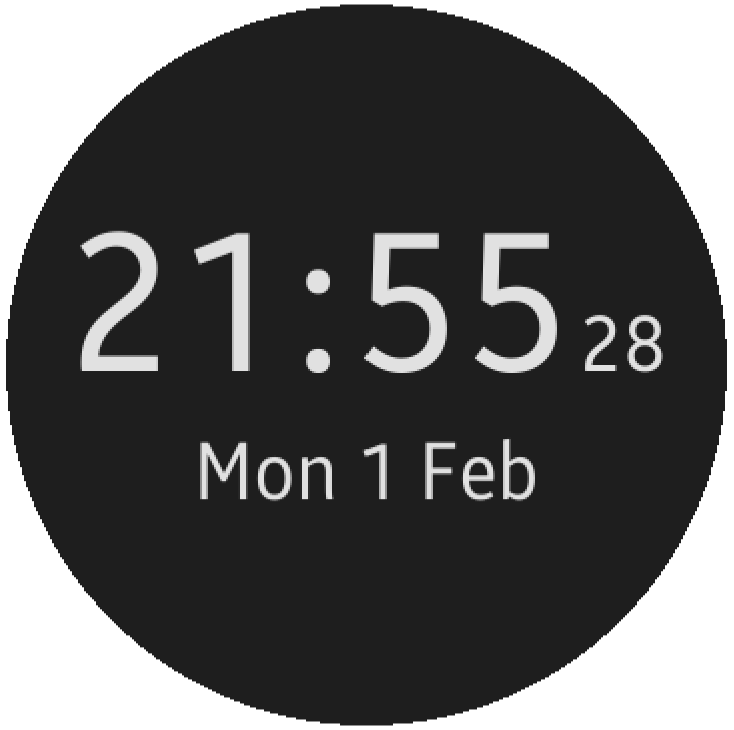


## Stage 3-1: Battery UI Layout – index.html

Add a new `<div>` element with a "`battery-container`" ID on top of the `<div>` element with a "`time-area`" ID, and add another `<div>` element with a "`battery-fill`" ID inside the new `<div>` element.

`index.html`

```
...
    <div id="main" class="page">
        <div id="battery-container">
            <div id="battery-fill"></div>
        </div>
        <div id="time-area" class="contents">
            <span id="hour-text">00</span>
            <span id="colon-text">:</span>
            <span id="minute-text">00</span>
            <span id="second-text">00</span>
        </div>
        <div id="date-area" class="contents">
            <span id="date-text">Mon 00 Jan</span>
        </div>
    </div>
...
```



At the moment, you cannot see any change on the screen because the `<div>` elements are empty. Next, add some styling to the `<div>` elements so that it appears properly as a battery element.

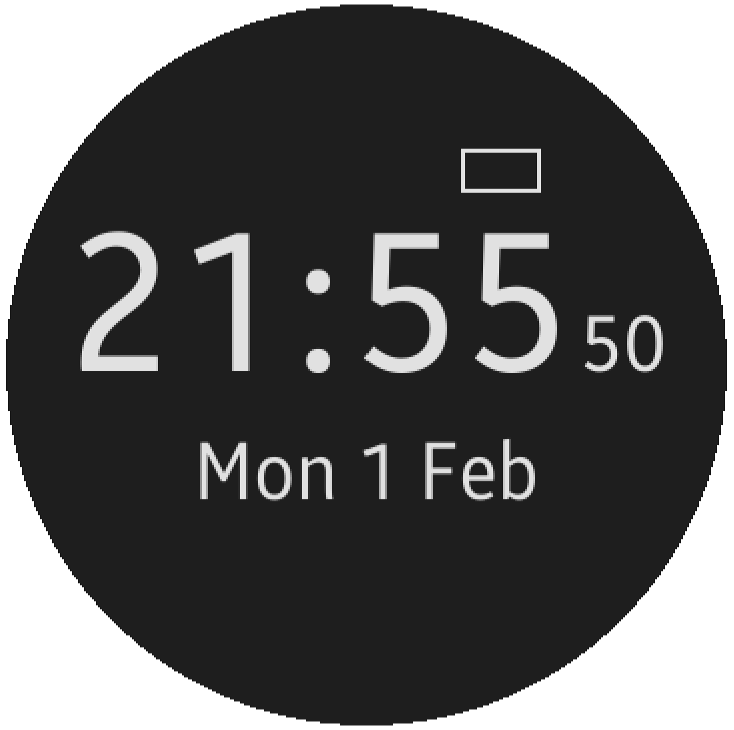
## Stage 3-2: Battery UI Layout – style.css

Add styling for the `<div>` element with the "`battery-container`" ID.

style.css

```
...

#colon-text, #minute-text {
    font-size: 100px;
}
#battery-container {
    position: absolute;
    width: 10%;
    height: 5%;
    top: 20%;
    left: 63%;
    border: solid 2px white;
}
```

Add styling for the "`#battery-container`".



To learn more about the CSS `position` property, see:
http://www.w3schools.com/css/css_positioning.asp
To learn more about the CSS `border` property, see:
http://www.w3schools.com/css/css_border.asp

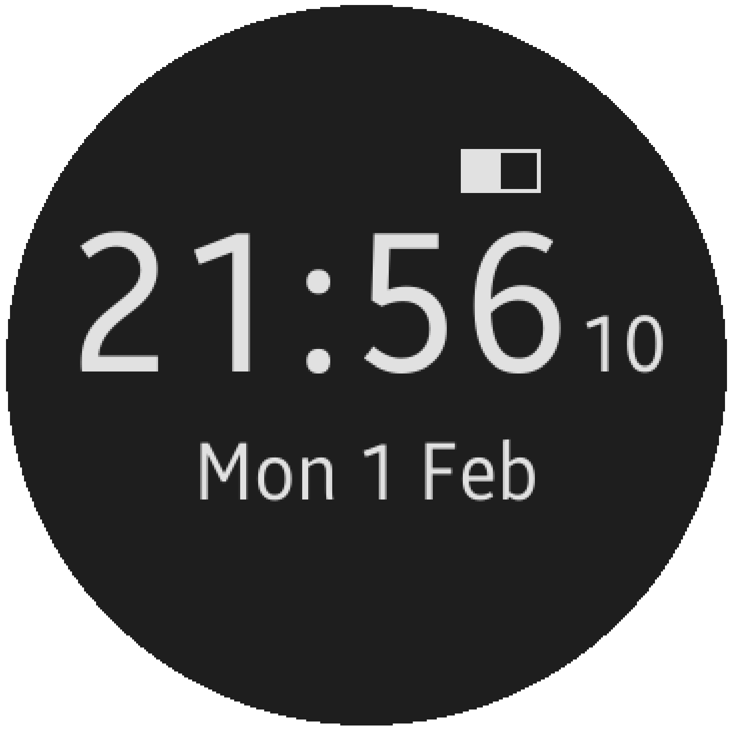
## Stage 3-2: Battery UI Layout – style.css

Also, add styling for the `<div>` element with the "`battery-fill`" ID.

style.css

```
...

#battery-container {
    position: absolute;
    width: 10%;
    height: 5%;
    top: 20%;
    left: 63%;
    border: solid 2px white;
}
#battery-fill {
    width: 50%;
    height: 100%;
    background-color: white;
}
```

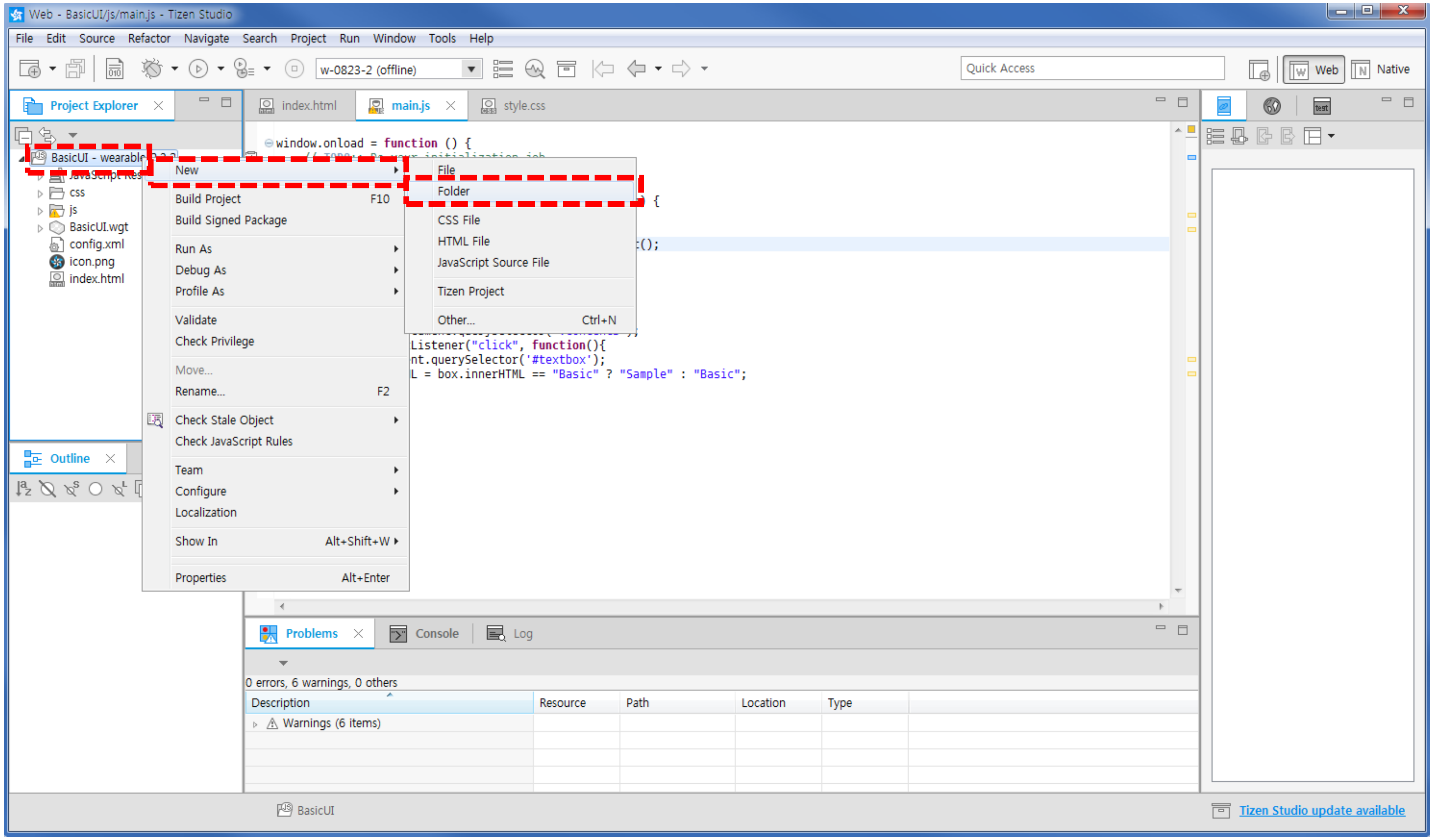
Add styling for the "`#battery-fill`".



To learn more about the CSS `background-color` property, see:
http://www.w3schools.com/cssref/pr_background-color.asp

# Stage 3-2: Battery UI Layout – Adding Files

- Finally, add the background image.
- To add a background image for the digital watch, copy the local image file to the project in the Tizen IDE.
- First, create a new folder by right-clicking the project name, and selecting **New > Folder**.



```
window.onload = function () {
                                    ) {
                                    ();
Listener("click", function(){
nt.querySelector('#textbox');
L = box.innerHTML == "Basic" ? "Sample" : "Basic";
```

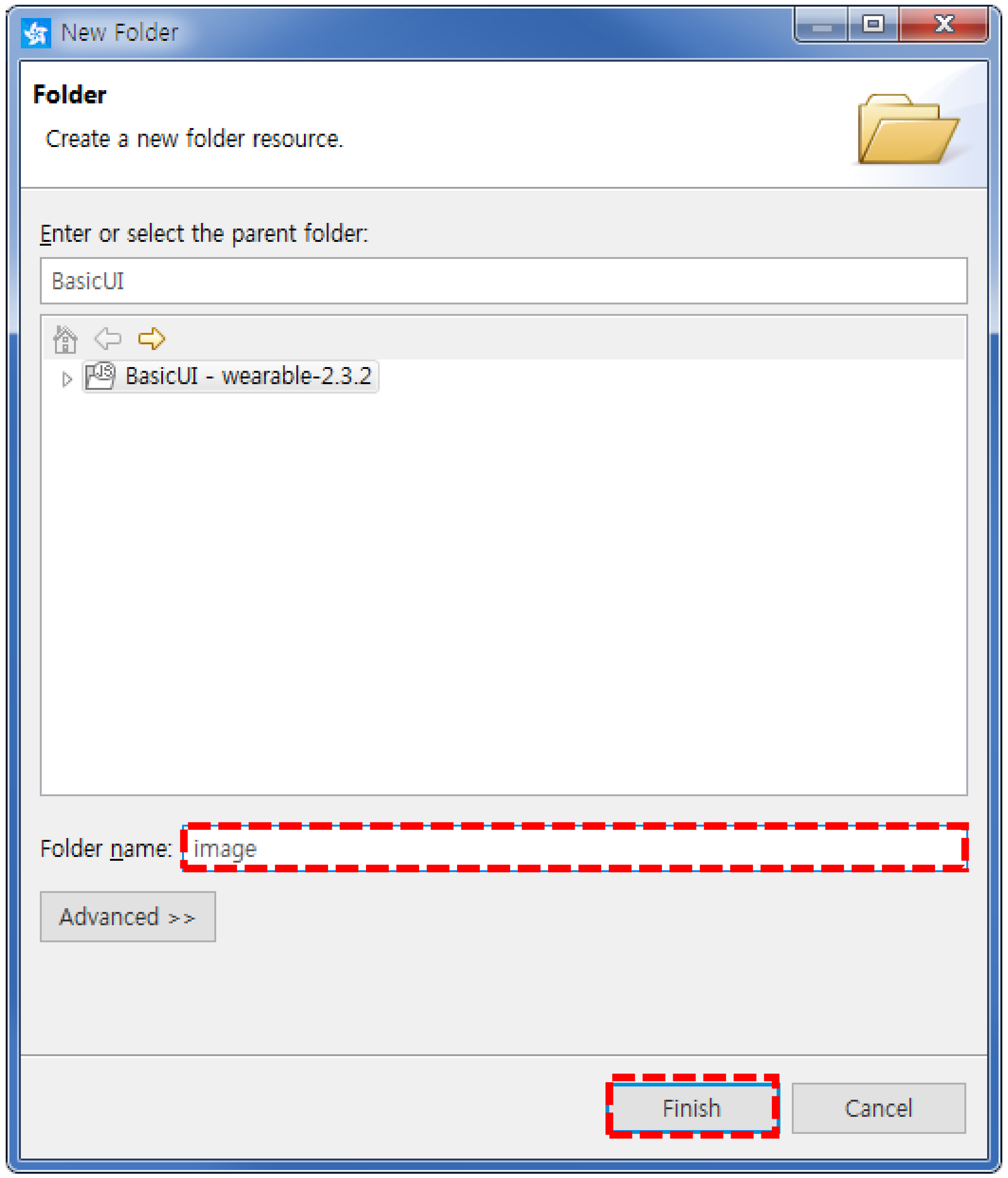
0 errors, 6 warnings, 0 others

Warnings (6 items)

## Stage 3-2: Battery UI Layout – Adding Files

When the **New Folder** pop-up appears, fill in the folder name as “**image**”, and click **Finish**.

The new folder has been created in the **Project Explorer**.



New Folder

**Folder**

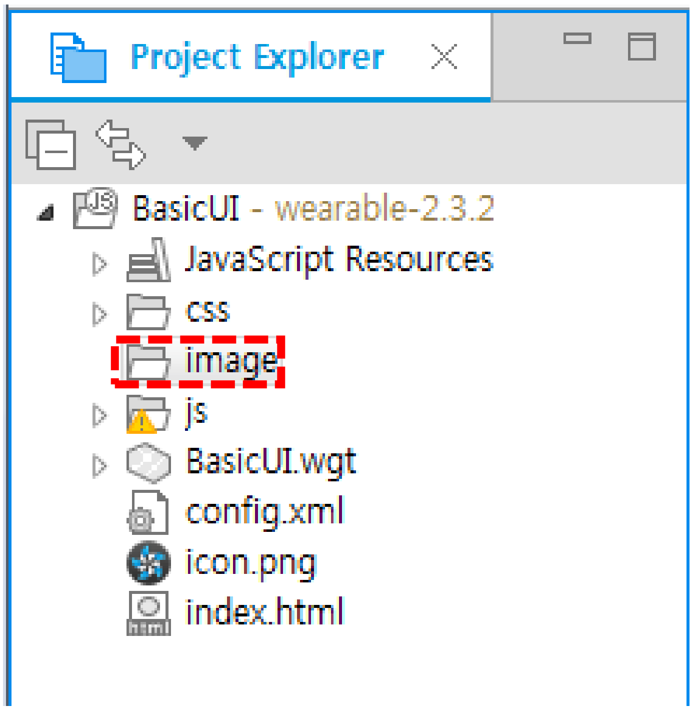
Create a new folder resource.

Enter or select the parent folder:

BasicUI

BasicUI - wearable-2.3.2

Folder name: image



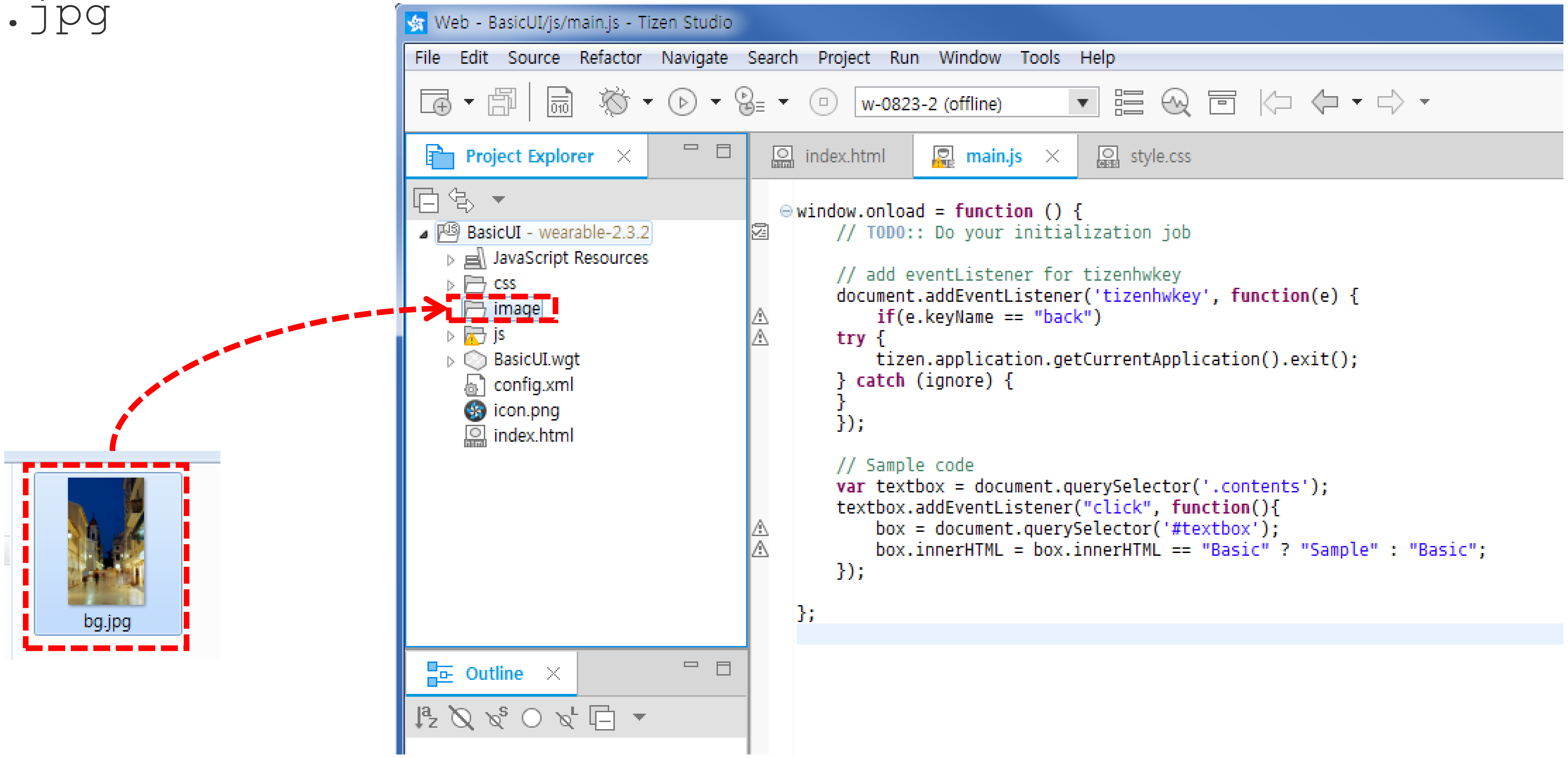
Project Explorer

- BasicUI - wearable-2.3.2
  - JavaScript Resources
  - css
  - image
  - js
  - BasicUI.wgt
  - config.xml
  - icon.png
  - index.html

## Stage 3-2: Battery UI Layout – Adding Files

Copy the image file from a local directory to the project folder by dragging and dropping the file to the **image** folder in **Project Explorer**.

You can use any image file from your local directory. Just rename the file as "`bg.jpg`"

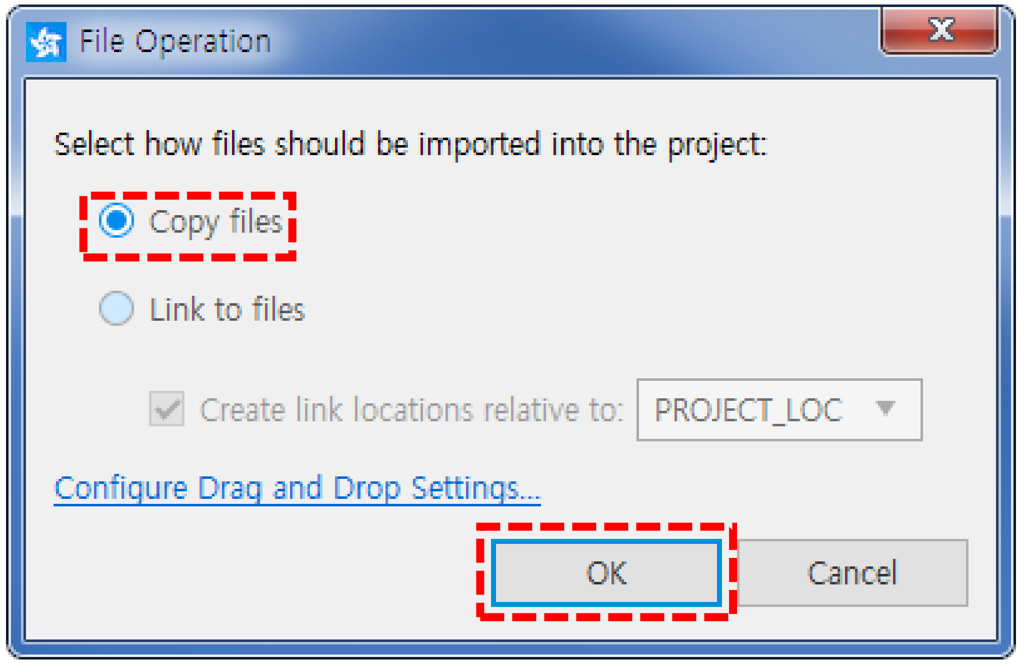


```
window.onload = function () {
    // TODO:: Do your initialization job

    // add eventListener for tizenhwkey
    document.addEventListener('tizenhwkey', function(e) {
        if(e.keyName == "back")
    try {
        tizen.application.getCurrentApplication().exit();
    } catch (ignore) {
    }
    });

    // Sample code
    var textbox = document.querySelector('.contents');
    textbox.addEventListener("click", function(){
        box = document.querySelector('#textbox');
        box.innerHTML = box.innerHTML == "Basic" ? "Sample" : "Basic";
    });

};
```

## Stage 3-2: Battery UI Layout – Adding Files

When the **File Operation** pop-up appears, select **Copy files**, and click **OK**.

The image file (`bg.jpg`) has been added in the **Project Explorer**.



File Operation

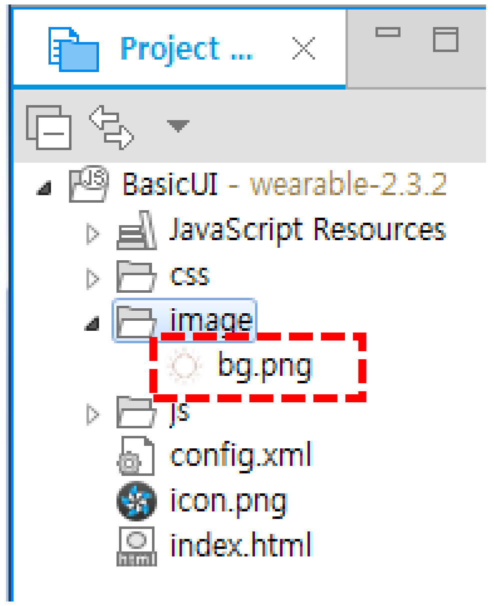
Select how files should be imported into the project:

Copy files [✓]

Link to files

Create link locations relative to: PROJECT_LOC

Configure Drag and Drop Settings...



## Stage 3-2: Battery UI Layout – style.css

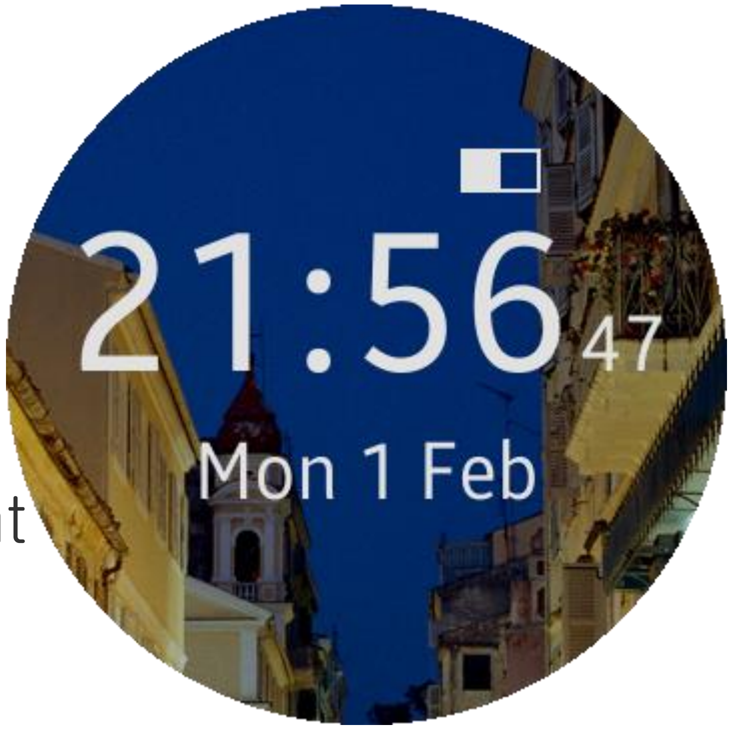
In the `style.css` file, add the background image for the **`html, body`** element.

style.css

```
html,
body {
    ...
    background-color: #222222;
    background-image: url('../image/bg.jpg');
    background-size: 100%;
    color: #ffffff;
}
...
```

Add the background image.

The Stage 3 is now completed, and the battery element and background image are implemented.

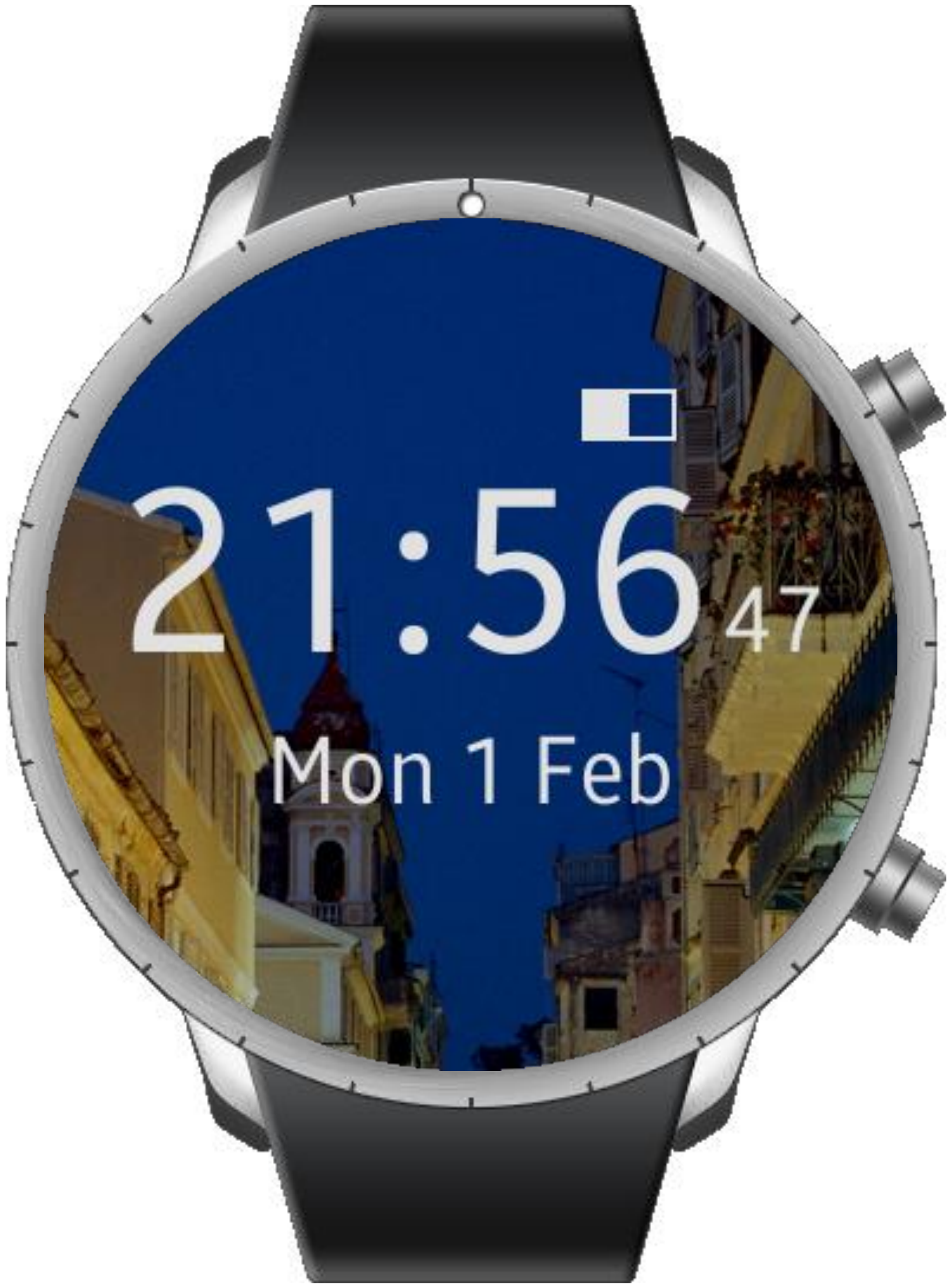


To learn more about the CSS `background-image` property, see: [http://www.w3schools.com/cssref/pr_background-image.asp](http://www.w3schools.com/cssref/pr_background-image.asp)

To learn more about the CSS `background-size` property, see: [http://www.w3schools.com/cssref/css3_pr_background-size.asp](http://www.w3schools.com/cssref/css3_pr_background-size.asp)

## Stage 3-3: Battery UI Layout – Summary

By the end of Stage 3, your application should appear as in the following figure:



## Stage 3-3: Battery UI Layout – Summary

Your `index.html` file should appear as in the following example:

index.html

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0, maximum-scale=1.0">
    <meta name="description" content="Tizen Wearable Web Basic Template" />

    <title>Tizen Wearable Web Basic Application</title>

    <link rel="stylesheet" type="text/css" href="css/style.css" />
    <script src="js/main.js"></script>
</head>
```

The code continues on the next page.

## Stage 3-3: Battery UI Layout – Summary

index.html

```
<body>
    <div id="main" class="page">
        <div id="battery-container">
            <div id="battery-fill"></div>
        </div>
        <div id="time-area" class="contents">
            <span id="hour-text">00</span>
            <span id="colon-text">:</span>
            <span id="minute-text">00</span>
            <span id="second-text">00</span>
        </div>
        <div id="date-area" class="contents">
            <span id="date-text">Mon 00 Jan</span>
        </div>
    </div>
</body>
</html>
```

# Stage 3-3: Battery UI Layout – Summary

Your `style.css` file should appear as in the following example:

style.css

```
html,
body {
    width: 100%;
    height: 100%;
    margin: 0 auto;
    padding: 0;
    background-color: #222222;
    background-image:
url('../image/bg.jpg');
    background-size: 100%;
    color: #ffffff;
}
.page {
    width: 100%;
    height: 100%;
    display: table;
}
.contents {
    display: block;
    vertical-align: middle;
    text-align: center;
    -webkit-tap-highlight-color:
transparent;
}
#hour-text {
    font-size: 100px;
}
```

The code continues on the right

```
#time-area {
    margin-top: 25%;
}

#second-text, #date-text {
    font-size: 40px;
}

#colon-text, #minute-text {
    font-size: 100px;
}
#battery-container {
    position: absolute;
    width: 10%;
    height: 5%;
    top: 20%;
    left: 63%;
    border: solid 2px white;
}

#battery-fill {
    width: 50%;
    height: 100%;
    background-color: white;
}
```

## Stage 3-3: Battery UI Layout – Summary

Lastly, your `main.js` file should appear as in the following example:

main.js

```
function changeHTML(elemId, newContent) {
    var htmlElem = document.getElementById(elemId);

    htmlElem.innerHTML = newContent;
}
```

The code continues on the next page.

## Stage 3-3: Battery UI Layout – Summary

main.js

```
function changeTime() {
    var date = new Date(),
        currentHour = date.getHours(),
        currentMinute = date.getMinutes(),
        currentSecond = date.getSeconds();

    // Make the hour, minute, and second texts show as double-digit numbers
    if (currentHour < 10) {
        changeHTML("hour-text", "0" + currentHour);
    } else {
        changeHTML("hour-text", currentHour);
    }

    if (currentMinute < 10) {
        changeHTML("minute-text", "0" + currentMinute);
    } else {
        changeHTML("minute-text", currentMinute);
    }

    if (currentSecond < 10) {
        changeHTML("second-text", "0" + currentSecond);
    } else {
        changeHTML("second-text", currentSecond);
    }
}
```

The code continues on the next page.

## Stage 3-3: Battery UI Layout – Summary

main.js

```
function changeDate() {
    var date = new Date(),
        currentMonth = date.getMonth(),
        currentDate = date.getDate(),
        currentDay = date.getDay(),
        strDate ="",
        arrayMonth = ["Jan", "Feb", "Mar", "Apr", "May", "Jun",
                      "Jul", "Aug", "Sep", "Oct", "Nov", "Dec" ],
        arrayDay = ["Sun", "Mon", "Tue", "Wed", "Thu", "Fri", "Sat"];

    currentMonth = arrayMonth[currentMonth];
    currentDay = arrayDay[currentDay];

    strDate = currentDay + " " + currentDate + " " + currentMonth;

    changeHTML("date-text", strDate);
}
```

The code continues on the next page.

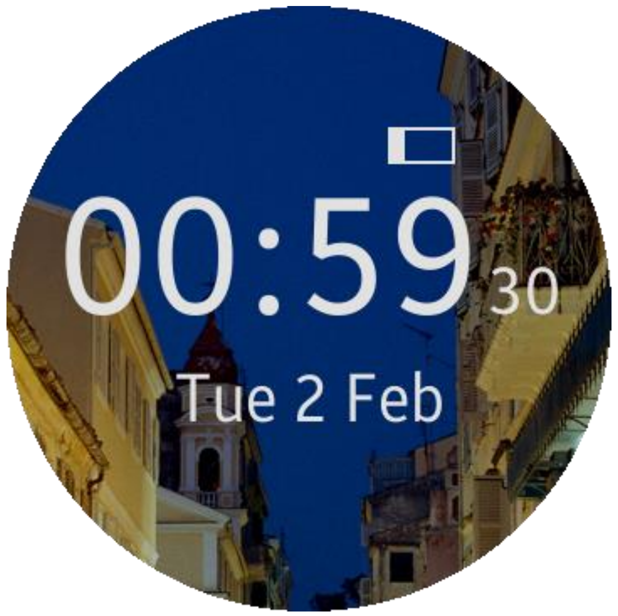
## Stage 3-3: Battery UI Layout – Summary

main.js

```
window.onload = function() {
    // TODO:: Do your initialization job
    setInterval(function() {
        changeTime();
    }, 1000);

    changeDate();

    // Add eventListener for tizenhwkey
    document.addEventListener('tizenhwkey', function(e) {
        if (e.keyName === "back") {
            try {
                tizen.application.getCurrentApplication().exit();
            } catch (ignore) {}
        }
    });
};
```

## Stage 4: Battery Functionality – Goal

The goal of Stage 4 is to implement the functionality of the battery indicator using JavaScript and the Tizen Device API.



In Stage 4-1, declare a privilege in the `config.xml` file to allow the Tizen System Information API to read system information.

In Stage 4-2, create a `changeBattery()` method using the Tizen System Information API.

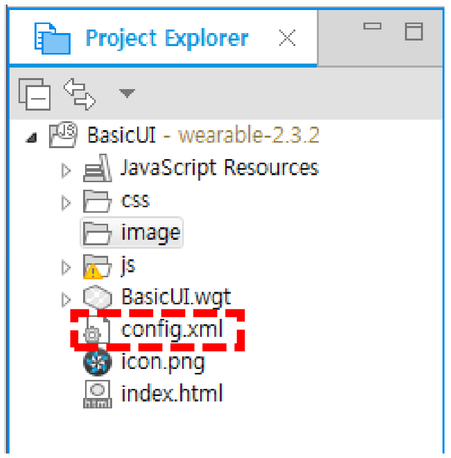
In Stage 4-3, the stage is summarized.

## Stage 4-1: Battery Functionality – config.xml

Tizen provides API-level access control for security-sensitive operations which, if not used correctly, can harm user privacy and system stability. Therefore, applications that use such sensitive APIs must declare the required privileges in the `config.xml` file.

To use the Tizen System Information API for getting the data on the battery level, first learn how to declare the privilege.

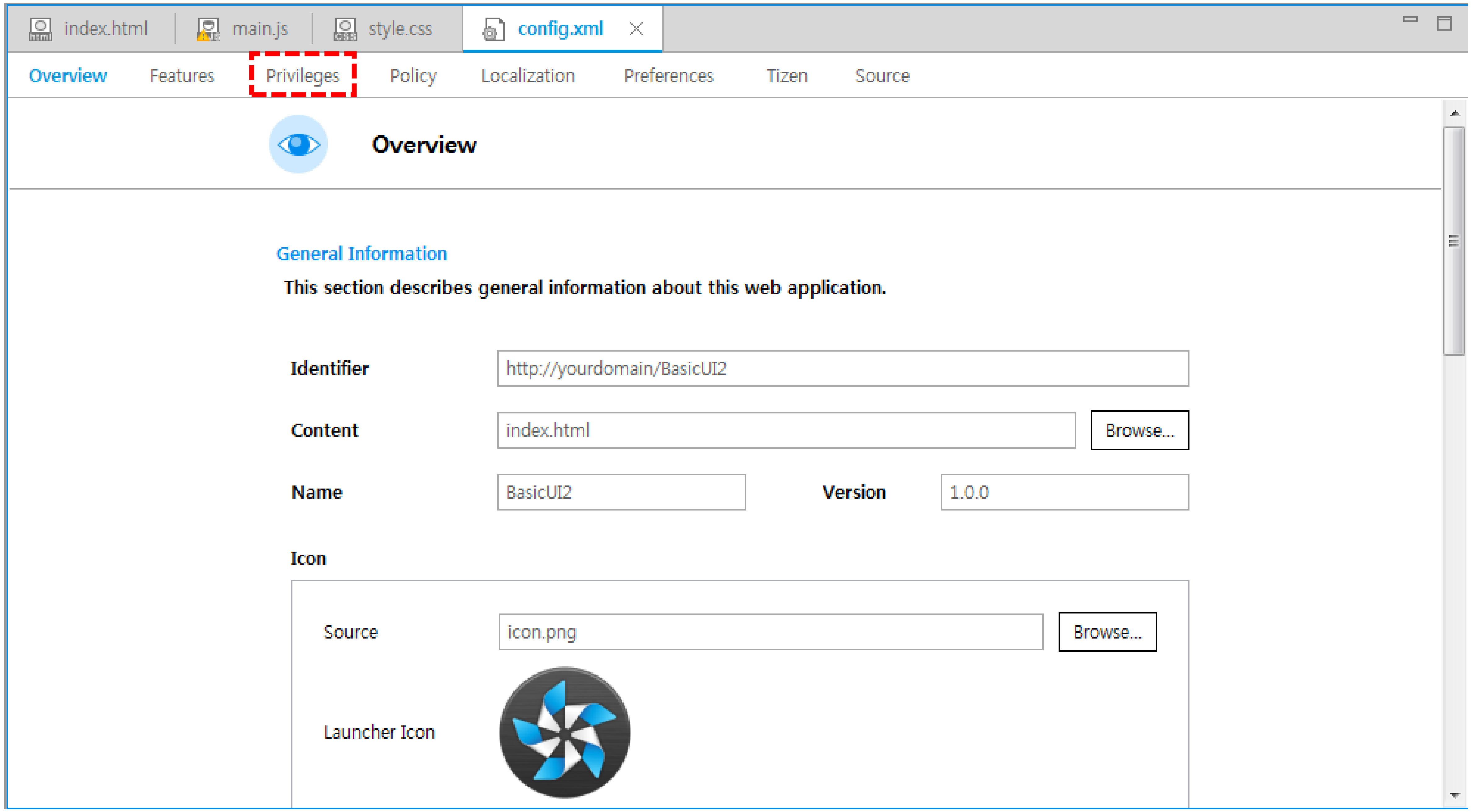
Open the `config.xml` file by double-clicking it in the **Project Explorer**.



To learn more about Tizen Security and API Privileges, see: [https://developer.tizen.org/development/getting-started/web-application/understanding-tizen-programming/security-and-api-privileges](https://developer.tizen.org/development/getting-started/web-application/understanding-tizen-programming/security-and-api-privileges)

## Stage 4-1: Battery Functionality – config.xml

Select the **Privileges** tab on the bottom of the `config.xml` page.



General Information

This section describes general information about this web application.

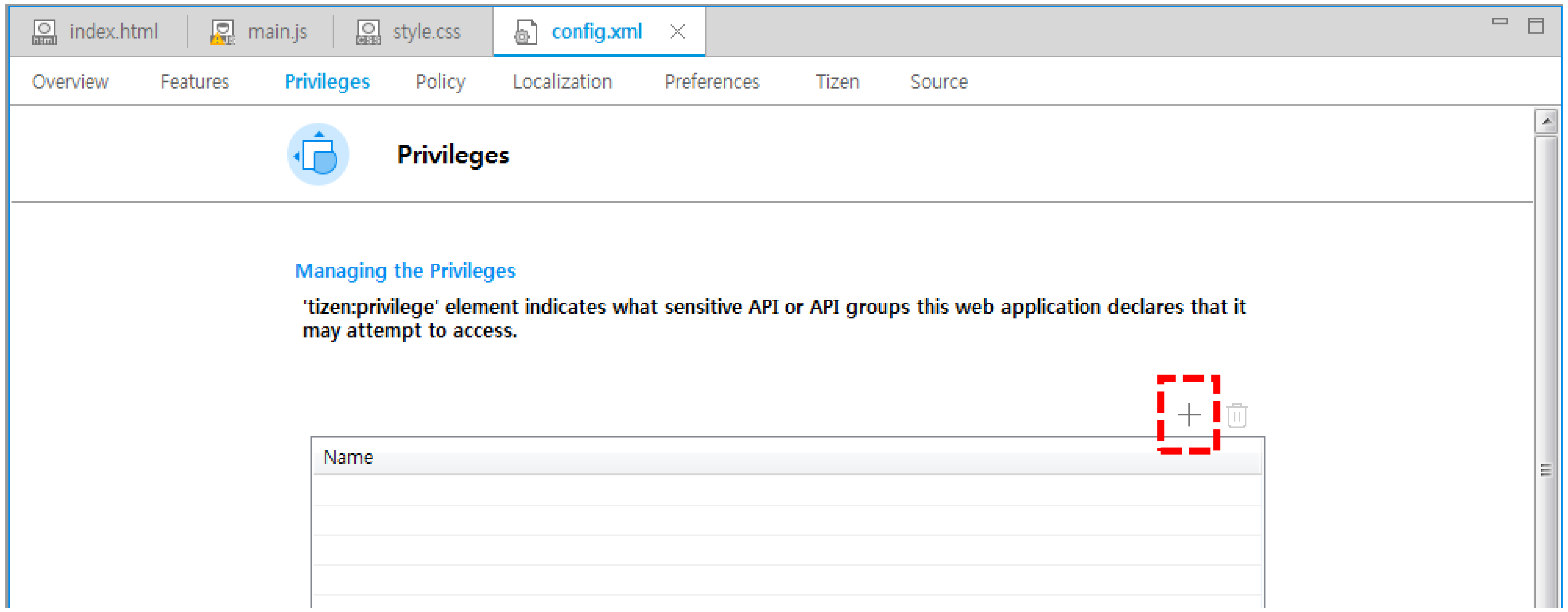
| | | | |
|---|---|---|---|
| Identifier | http://yourdomain/BasicUI2 | | |
| Content | index.html | | |
| Name | BasicUI2 | Version | 1.0.0 |

Icon

| | |
|---|---|
| Source | icon.png |
| Launcher Icon | |

# Stage 4-1: Battery Functionality – config.xml

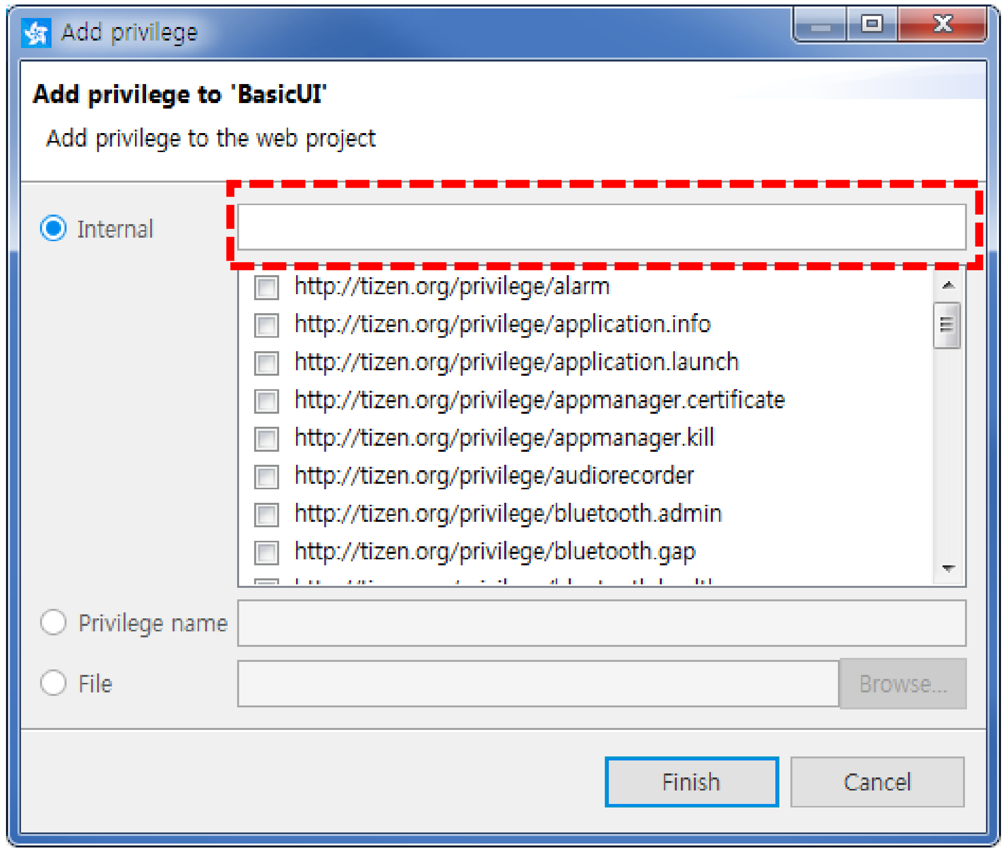
In the **Privileges** tab, click **+** (Add).



Managing the Privileges

'tizen:privilege' element indicates what sensitive API or API groups this web application declares that it may attempt to access.

In the **Add privilege** pop-up, select the textbox for **Internal**.



Add privilege to 'BasicUI'

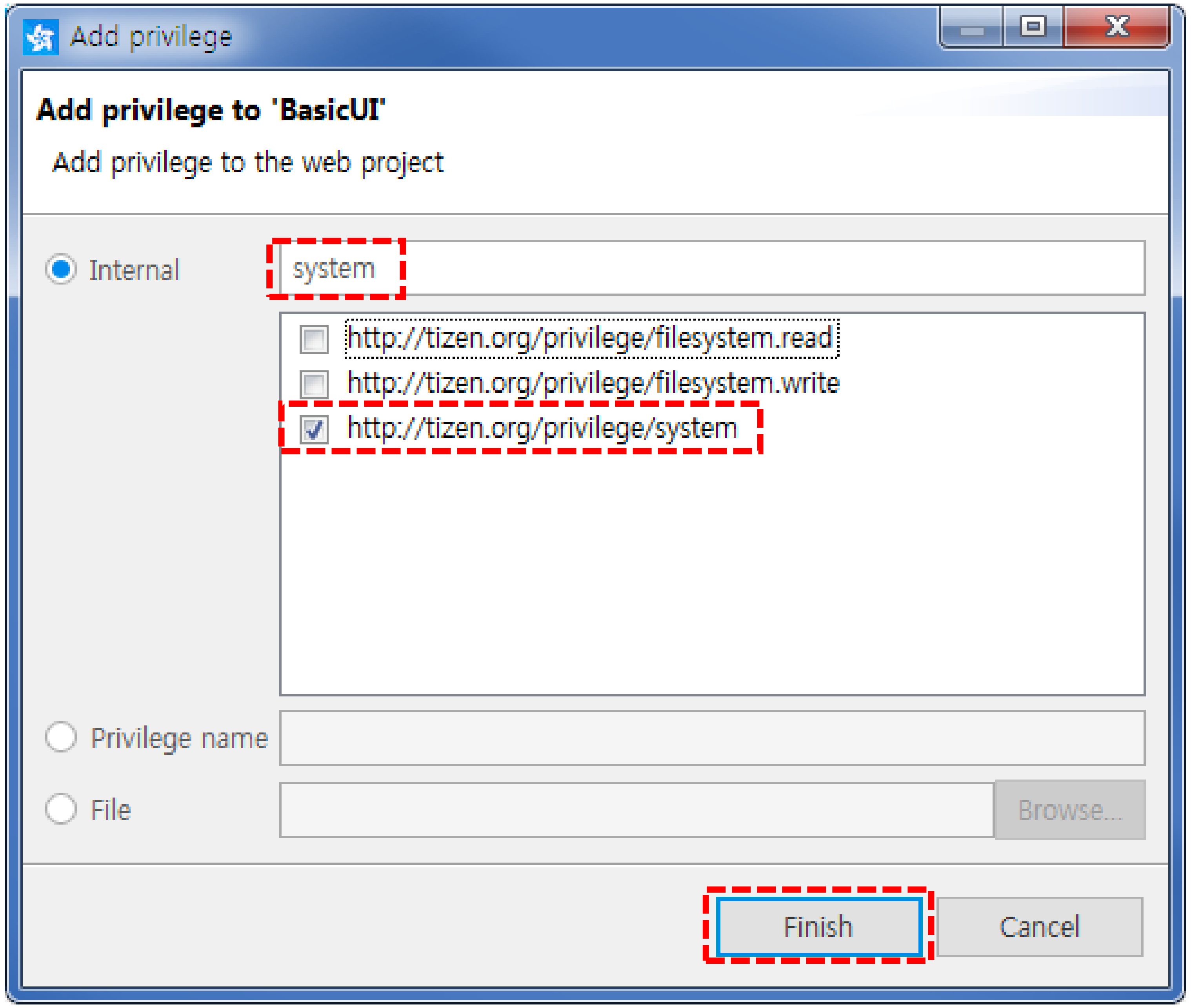
Add privilege to the web project

Internal

- http://tizen.org/privilege/alarm
- http://tizen.org/privilege/application.info
- http://tizen.org/privilege/application.launch
- http://tizen.org/privilege/appmanager.certificate
- http://tizen.org/privilege/appmanager.kill
- http://tizen.org/privilege/audiorecorder
- http://tizen.org/privilege/bluetooth.admin
- http://tizen.org/privilege/bluetooth.gap

# Stage 4-1: Battery Functionality – config.xml

Enter "system" in the textbox, select `http://tizen.org/privilege/system` in the list, and click **Finish**.



Add privilege

**Add privilege to 'BasicUI'**

Add privilege to the web project

(●) Internal: system

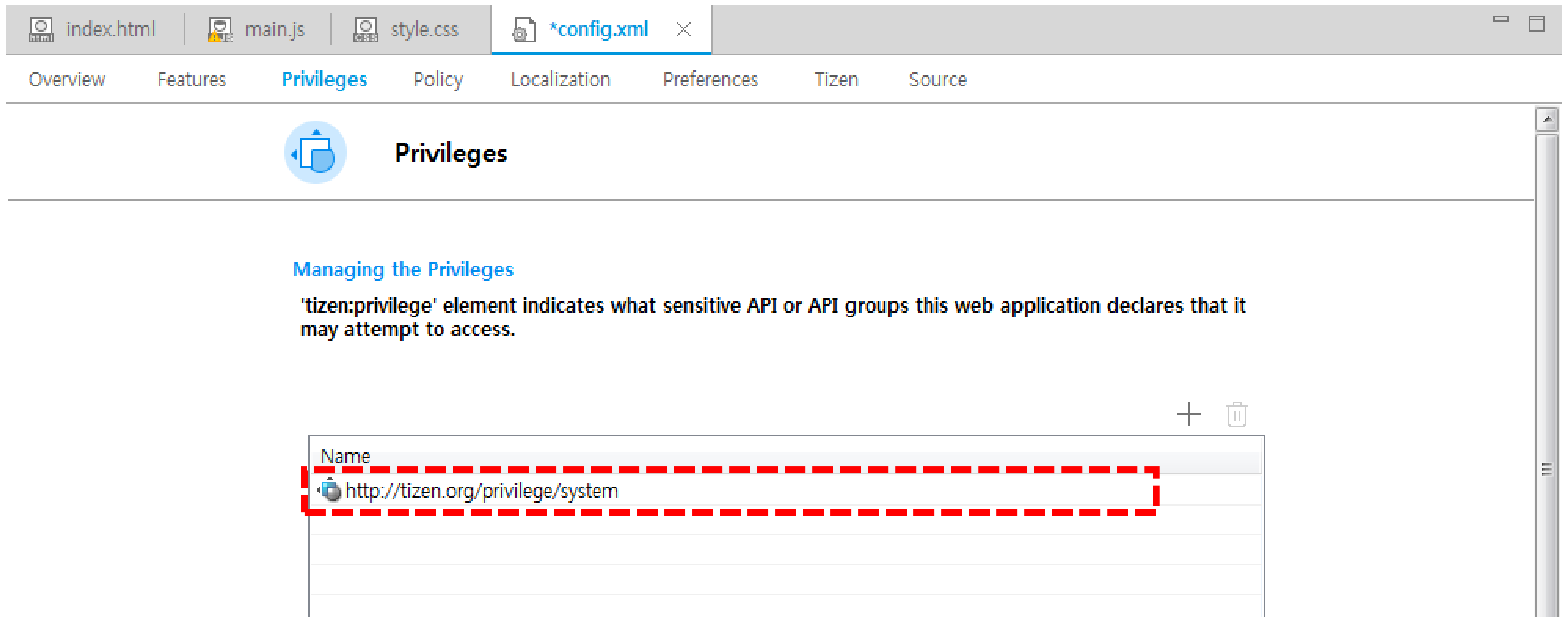
- [ ] http://tizen.org/privilege/filesystem.read
- [ ] http://tizen.org/privilege/filesystem.write
- [x] http://tizen.org/privilege/system

( ) Privilege name

( ) File

# Stage 4-1: Battery Functionality – config.xml

The `http://tizen.org/privilege/system` privilege has been added.



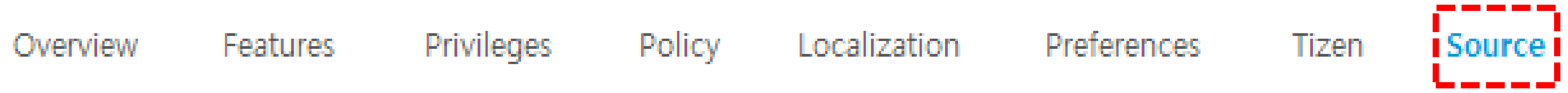
Managing the Privileges

'tizen:privilege' element indicates what sensitive API or API groups this web application declares that it may attempt to access.

| Name |
|---|
| http://tizen.org/privilege/system |

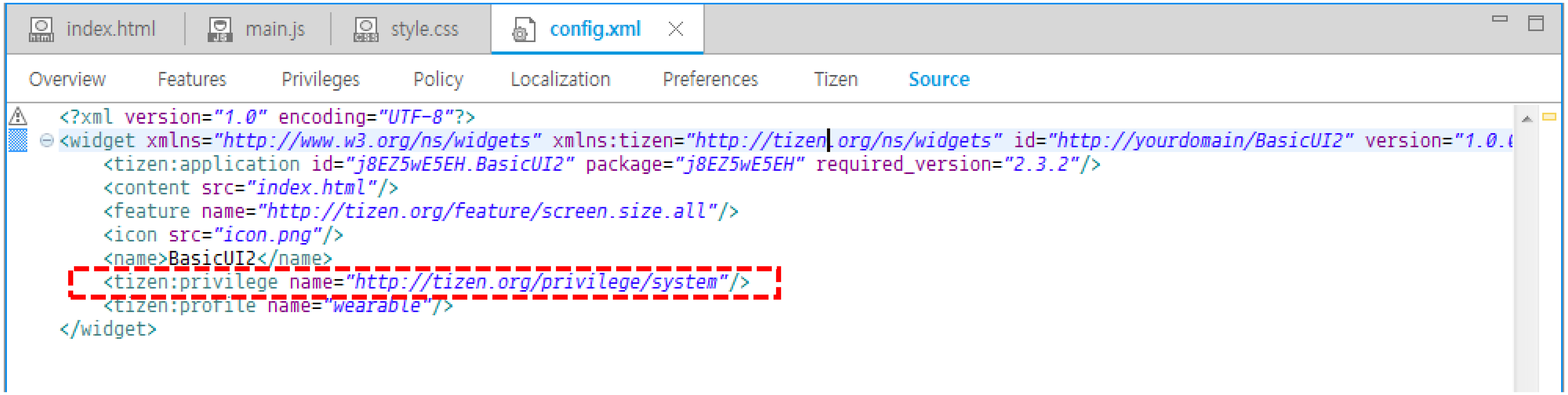
Press **Ctrl** + **S** to save the change in the `config.xml` file.

You can double-check the change in the source code.
Select the **Source** tab on the bottom of the `config.xml` page.



## Stage 4-1: Battery Functionality – config.xml

A new line for `tizen:privilege` has been added in the `config.xml` file.



```xml
<?xml version="1.0" encoding="UTF-8"?>
<widget xmlns="http://www.w3.org/ns/widgets" xmlns:tizen="http://tizen.org/ns/widgets" id="http://yourdomain/BasicUI2" version="1.0.(
    <tizen:application id="j8EZ5wE5EH.BasicUI2" package="j8EZ5wE5EH" required_version="2.3.2"/>
    <content src="index.html"/>
    <feature name="http://tizen.org/feature/screen.size.all"/>
    <icon src="icon.png"/>
    <name>BasicUI2</name>
    <tizen:privilege name="http://tizen.org/privilege/system"/>
    <tizen:profile name="wearable"/>
</widget>
```

You are now ready to use the Tizen System Information API.

## Stage 4-2: Battery Functionality – changeBattery()

Implement a function which gets the battery level of the system using the Tizen System Information API and changes the battery indicator accordingly.

main.js

```
...
function changeDate() {
    ...
}

function changeBattery() {
   function onSuccessCallback(battery) {
       document.getElementById("battery-fill").style.width = (battery.level * 100) + "%";
   }
   function onErrorCallback(error) {
       alert("Not supported: " + error.message);
   }
   tizen.systeminfo.addPropertyValueChangeListener("BATTERY", onSuccessCallback,
onErrorCallback);
}

window.onload = function() {
    ...
};
```

To learn more about the Tizen System Information API, see: https://developer.tizen.org/community/tip-tech/system-information-api-guide

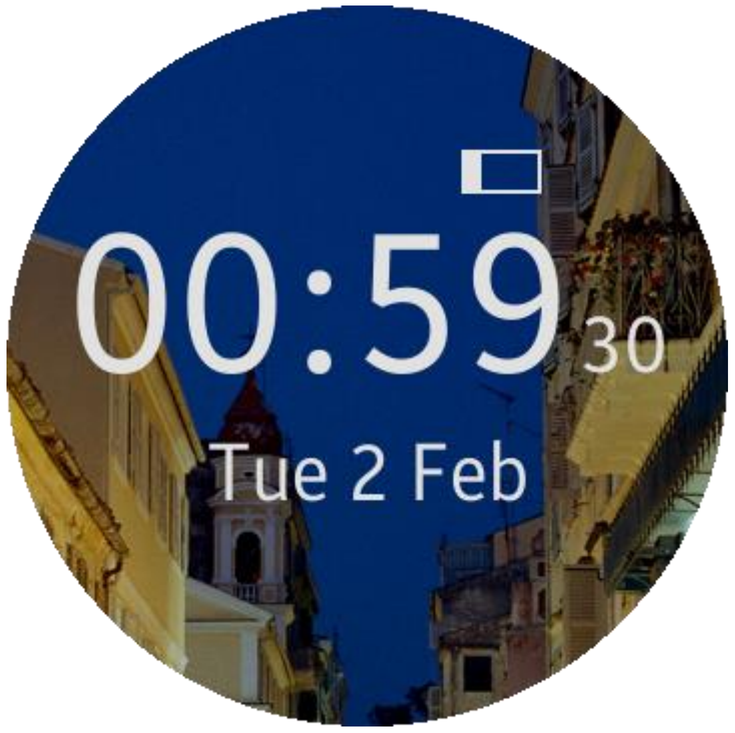
## Stage 4-2: Battery Functionality – changeBattery()

Call the `changeBattery()` method in the `window.onload` method so that it gets executed when the application is launched.

main.js

```
...

window.onload = function() {
    // TODO:: Do your initialization job
    setInterval(function() {
        changeTime();
    }, 1000);

    changeDate();
    changeBattery();
    ...
};
```

Call the `changeBattery` method when the window is loaded.

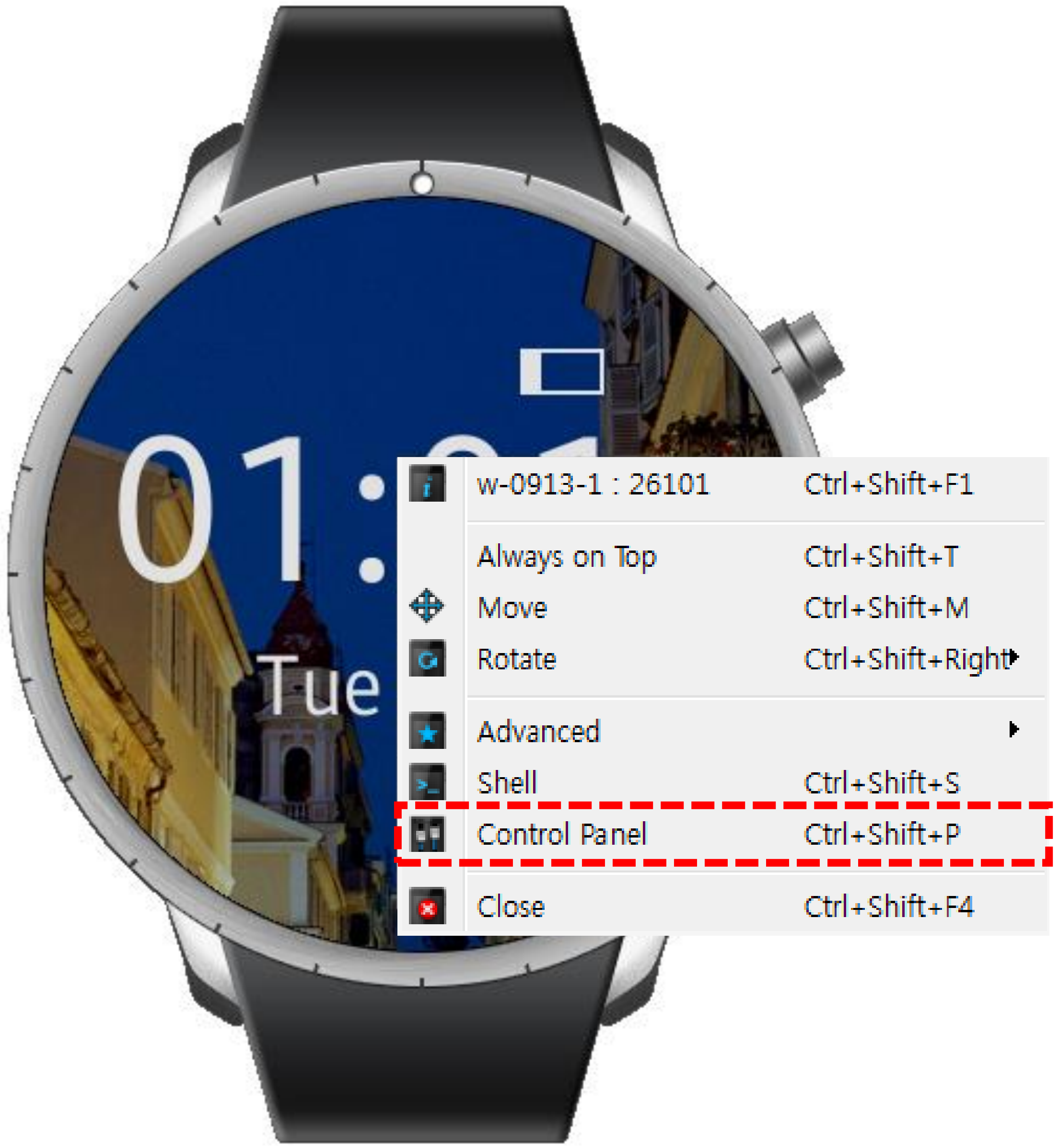


You can use the **Event Injector** in the **Emulator Control Panel** to artificially create and use any data, including the battery state and level, required during application execution.

Check if the `changeBattery()` method is working properly.

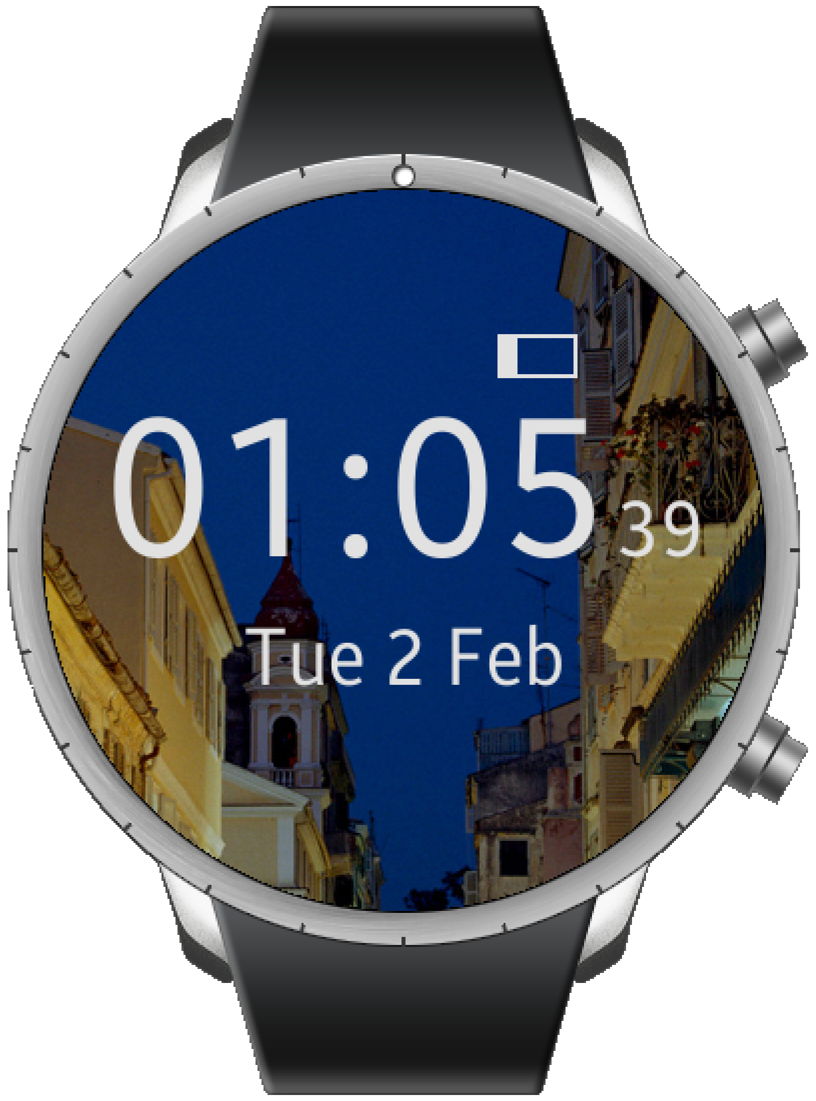
## Stage 4-2: Battery Functionality – changeBattery()

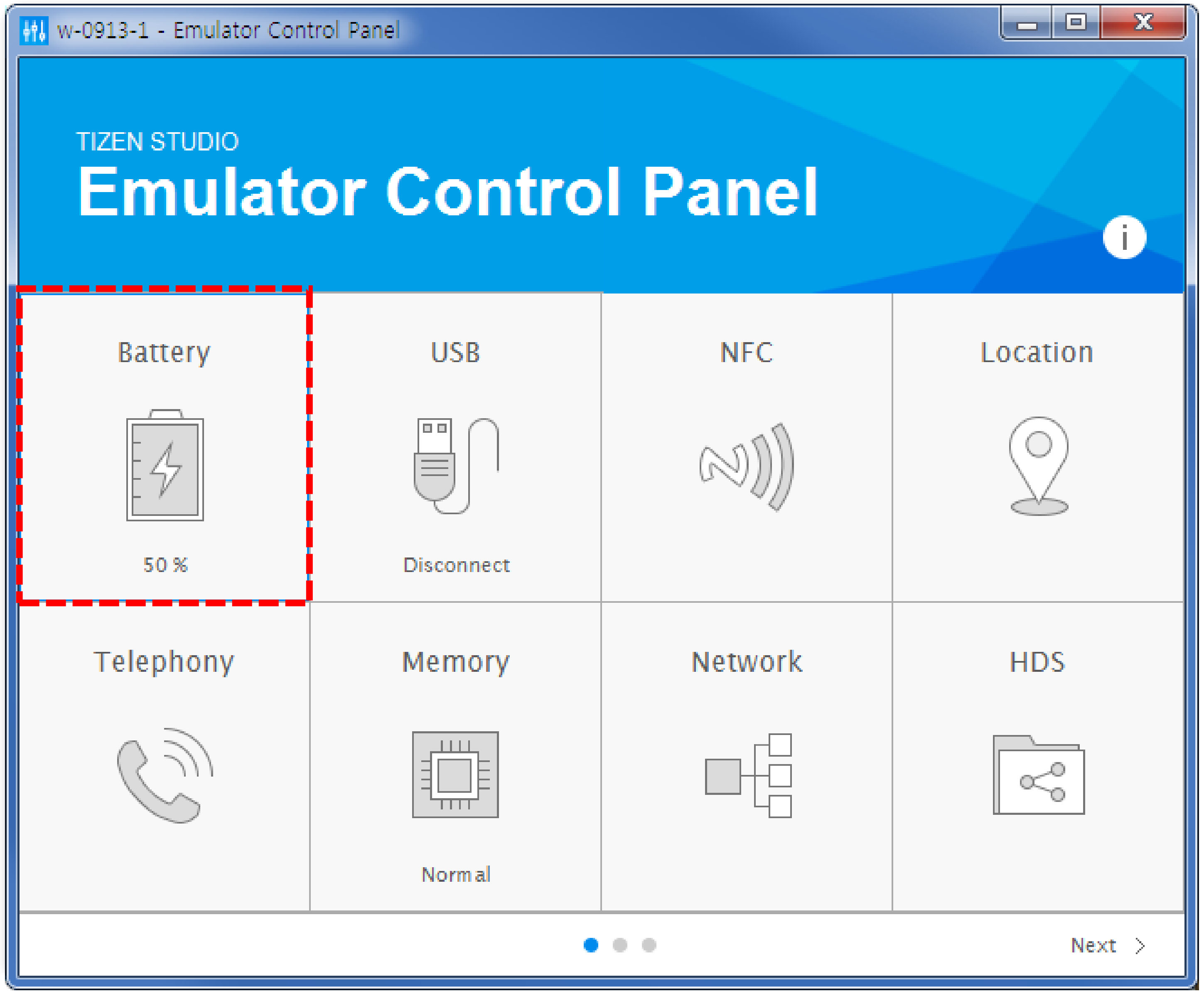
Right-click on the emulator screen to view more options, and click **Control Panel**.



## Stage 4-2: Battery Functionality – changeBattery()

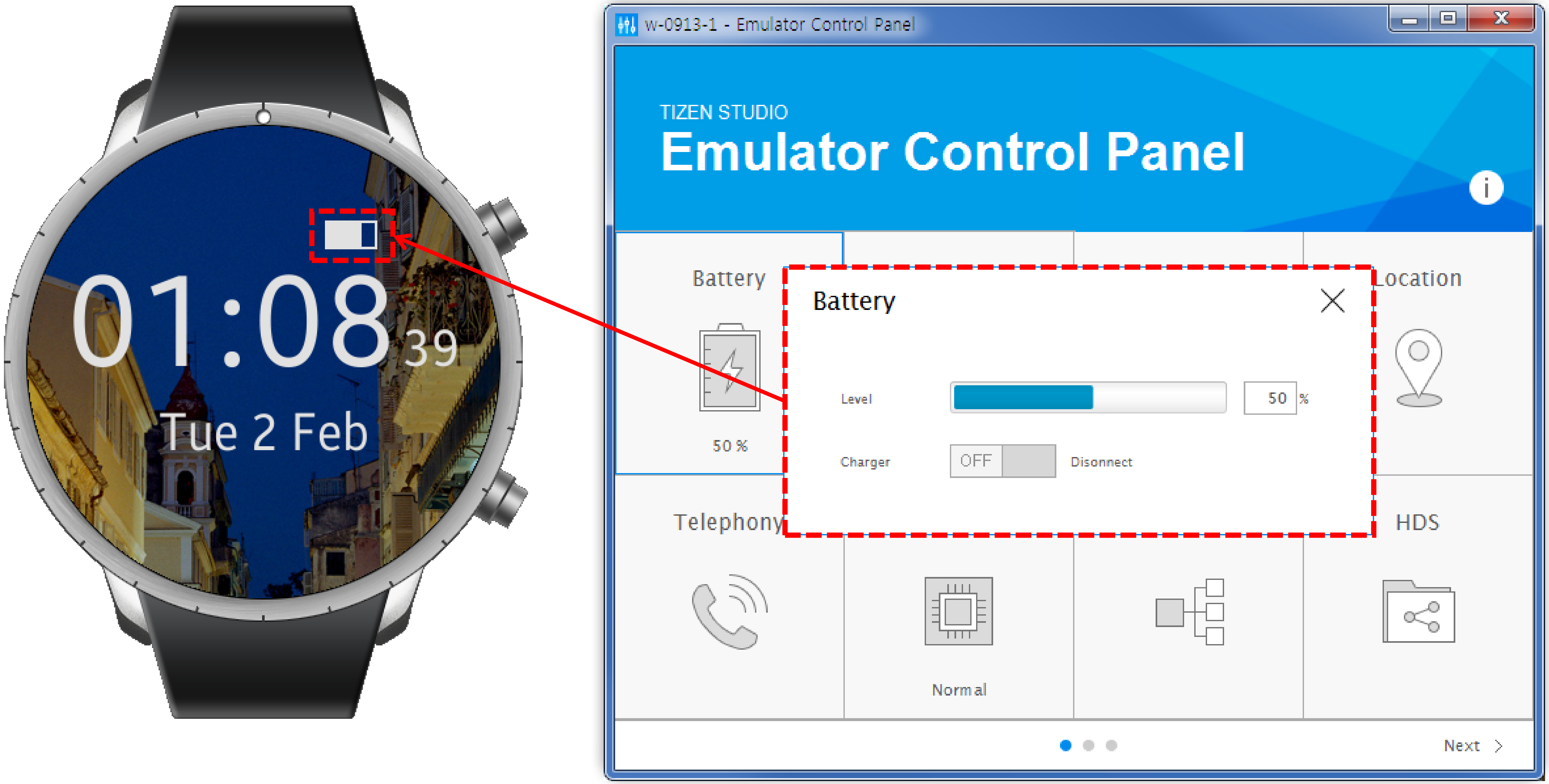
In the **Emulator Control Panel**, select **Battery** under **Event Injector** Category.





## Stage 4-2: Battery Functionality – changeBattery()

Try adjusting the battery level, and confirm that the battery indicator of the digital watch application changes accordingly.



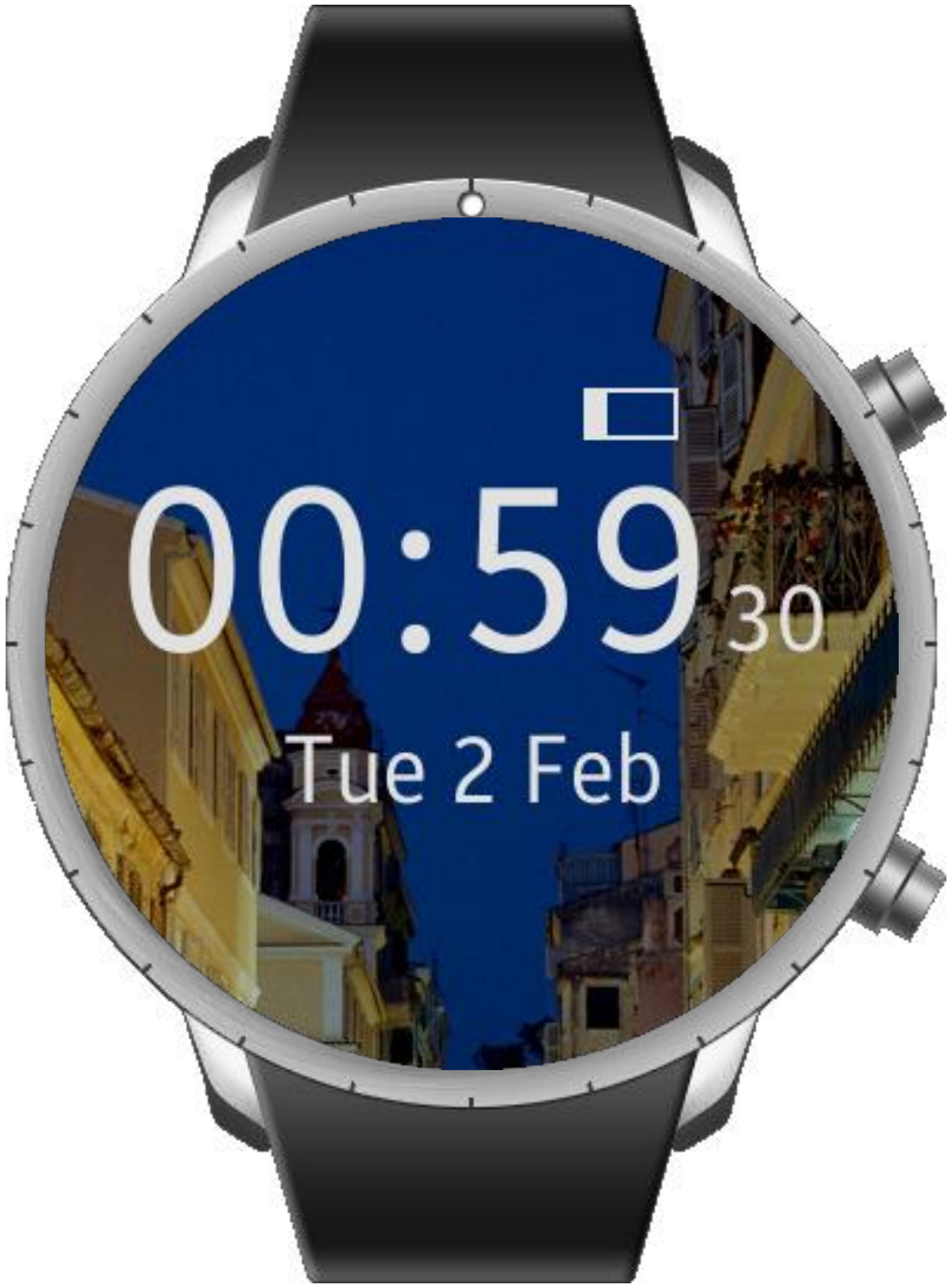
Battery

| | | |
|---|---|---|
| Level | 50 % | |
| Charger | OFF | Disconnect |

The Stage 4 is completed and the battery functionality is implemented.

## Stage 4-3: Battery Functionality – Summary

By the end of Stage 4, your application should appear as in the following figure:



## Stage 4-3: Battery Functionality – Summary

Your `index.html` file should appear as in the following example:

`index.html`

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0, maximum-
scale=1.0">
    <meta name="description" content="Tizen Wearable Web Basic Template" />

    <title>Tizen Wearable Web Basic Application</title>

    <link rel="stylesheet" type="text/css" href="css/style.css" />
    <script src="js/main.js"></script>
</head>
```

The code continues on the next page.

## Stage 4-3: Battery Functionality – Summary

index.html

```
<body>
    <div id="main" class="page">
        <div id="battery-container">
            <div id="battery-fill"></div>
        </div>
        <div id="time-area" class="contents">
            <span id="hour-text">00</span>
            <span id="colon-text">:</span>
            <span id="minute-text">00</span>
            <span id="second-text">00</span>
        </div>
        <div id="date-area" class="contents">
            <span id="date-text">Mon 00 Jan</span>
        </div>
    </div>
</body>
</html>
```

## Stage 4-3: Battery Functionality – Summary

Your `style.css` file should appear as in the following example:

`style.css`

```
html,
body {
    width: 100%;
    height: 100%;
    margin: 0 auto;
    padding: 0;
    background-color: #222222;
    background-image:
url('../image/bg.jpg');
    background-size: 100%;
    color: #ffffff;
}
.page {
    width: 100%;
    height: 100%;
    display: table;
}
.contents {
    display: block;
    vertical-align: middle;
    text-align: center;
    -webkit-tap-highlight-color:
transparent;
}
#hour-text {
    font-size: 100px;
}
```

The code continues on the right.

```
#time-area {
    margin-top: 25%;
}

#second-text, #date-text {
    font-size: 40px;
}

#colon-text, #minute-text {
    font-size: 100px;
}
#battery-container {
    position: absolute;
    width: 10%;
    height: 5%;
    top: 20%;
    left: 63%;
    border: solid 2px white;
}

#battery-fill {
    width: 50%;
    height: 100%;
    background-color: white;
}
```

## Stage 4-3: Battery Functionality – Summary

Lastly, your `main.js` file should appear as in the following example:

main.js
```
function changeHTML(elemId, newContent) {
    var htmlElem = document.getElementById(elemId);

    htmlElem.innerHTML = newContent;
}
```

The code continues on the next page.

## Stage 4-3: Battery Functionality – Summary

main.js

```
function changeTime() {
    var date = new Date(),
        currentHour = date.getHours(),
        currentMinute = date.getMinutes(),
        currentSecond = date.getSeconds();

    // Make the hour, minute, and second texts show as double-digit numbers
    if (currentHour < 10) {
        changeHTML("hour-text", "0" + currentHour);
    } else {
        changeHTML("hour-text", currentHour);
    }

    if (currentMinute < 10) {
        changeHTML("minute-text", "0" + currentMinute);
    } else {
        changeHTML("minute-text", currentMinute);
    }

    if (currentSecond < 10) {
        changeHTML("second-text", "0" + currentSecond);
    } else {
        changeHTML("second-text", currentSecond);
    }
}
```

The code continues on the next page.

## Stage 4-3: Battery Functionality – Summary

main.js

```
function changeDate() {
    var date = new Date(),
        currentMonth = date.getMonth(),
        currentDate = date.getDate(),
        currentDay = date.getDay(),
        strDate ="",
        arrayMonth = ["Jan", "Feb", "Mar", "Apr", "May", "Jun",
                      "Jul", "Aug", "Sep", "Oct", "Nov", "Dec" ],
        arrayDay = ["Sun", "Mon", "Tue", "Wed", "Thu", "Fri", "Sat"];

    currentMonth = arrayMonth[currentMonth];
    currentDay = arrayDay[currentDay];

    strDate = currentDay + " " + currentDate + " " + currentMonth;

    changeHTML("date-text", strDate);
}
```

The code continues on the next page.

# Stage 4-3: Battery Functionality – Summary

main.js

```
function changeBattery() {
    function onSuccessCallback(battery) {
        document.getElementById("battery-fill").style.width = (battery.level * 100) + "%";
    }

    function onErrorCallback(error) {
        alert("Not supported: " + error.message);
    }
    tizen.systeminfo.addPropertyValueChangeListener("BATTERY", onSuccessCallback,
onErrorCallback);
}
```

The code continues on the next page.

## Stage 4-3: Battery Functionality – Summary

main.js

```
window.onload = function() {
    // TODO:: Do your initialization job
    setInterval(function() {
        changeTime();
    }, 1000);

    changeDate();

    // Add eventListener for tizenhwkey
    document.addEventListener('tizenhwkey', function(e) {
        if (e.keyName === "back") {
            try {
                tizen.application.getCurrentApplication().exit();
            } catch (ignore) {}
        }
    });
};
```