HANDLEBARS 1

WHAT IS HANDLEBARS

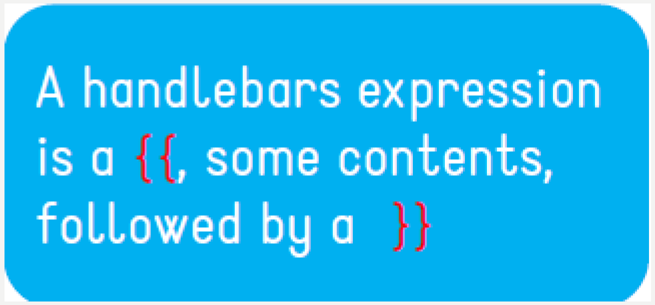
- Handlebars is a templating engine for JavaScript.
- It is a JavaScript library that you include in your page just as you include any other JavaScript file.
- With it, you can add templates to your HTML page that will be parsed and interpolated (values of properties inserted in place) with the values from the data (JSON object) you passed to the Handlebars function.

WHY HANDLEBARS

- We are building apps, not web sites
- We want to separate presentation from logic
- We don’t want to put any HTML element into JavaScript code

EXAMPLE OF TEMPLATE

```
<div class="userEntry">
 <h1>
  {{username}}
 </h1>
 <img>
  {{profilePic}}
 </img>
 <div class="status">
  {{status}}
 </div>
</div>
```



A handlebars expression is a {{, some contents, followed by a }}

VALUES ESCAPING

- Handlebars HTML-escapes all the values returned by an {{expression}}
- If you don't want Handlebars to escape a value, use the "triple-stash"

```
<div class="userEntry">
  <h1>{{username}}</h1>
  <div class="status">
    {{{status}}}
  </div>
 </div>
```

TEMPLATE CONTEXT

- A context is a Javascript object used to populate a template

```
var context = {
 username: "Ivano",
 profilePic: "./images/pic.png",
 status: "feeling good"
};
```

COMPILING A TEMPLATE

- Templates are defined within a <script> tag or in external files

```
<script id="user-template"
 type="text/x-handlebars-template">
 // template content
</script>
```

COMPILING A TEMPLATE

- Handlebars.compile is used to compile a template

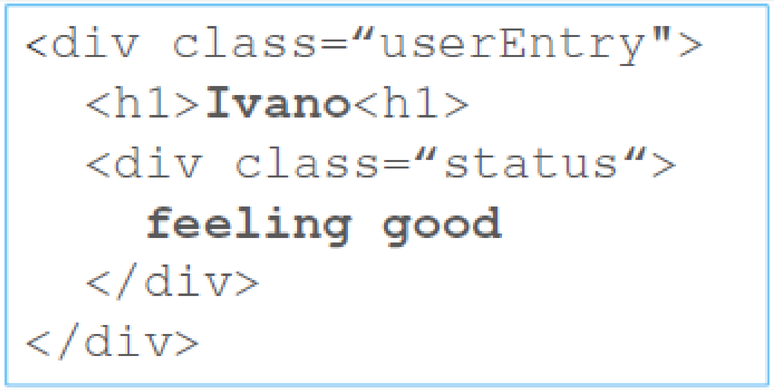
```
var source = $("#user-template").html();
var template = Handlebars.compile(source);
```

- Compiling = obtaining a JS object representing the template

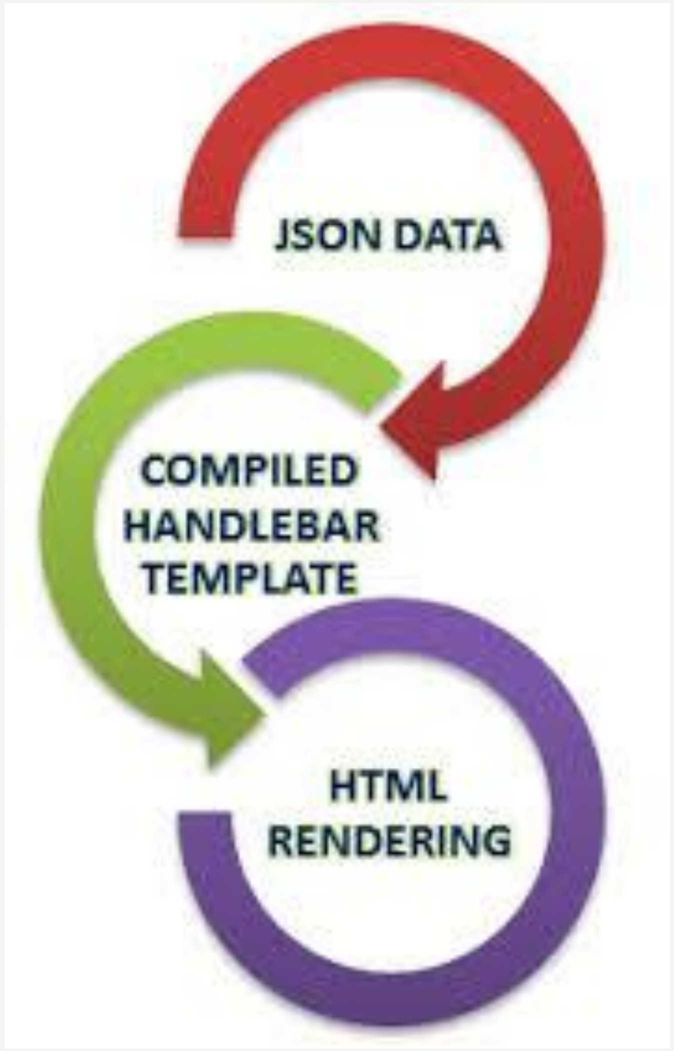
OBTAINING THE FINAL HTML CODE

- You have to execute a template with a context in order to get its corresponding HTML code

```
var context = {
 username: "Ivano",
 status: "feeling good"
};
var html =
 template(context);
```



```
<div class="userEntry">
  <h1>Ivano<h1>
  <div class="status">
    feeling good
  </div>
</div>
```


JSON DATA
COMPILED HANDLEBAR TEMPLATE
HTML RENDERING

EXPRESSIONS

- The simplest expression is a simple identifier

```
<h1>{{title}}</h1>
```

- This expression means "look up the title property in the current context"

EXPRESSIONS

- Handlebars expressions can also be dot-separated paths

```
<h1>{{user.username}}</h1>
```

- This expression means "look up the user property in the current context, then look up the username property in the result"

BUILT-IN HELPERS

- **Each**
  - To iterate over a list inside the block, you can use *this* to reference the element being iterated

```
{ people: [ "Yehuda Katz", "Alan Johnson", "Charles Jolley" ] }
```

```
<ul class="people_list">
 {{#each people}}
 <li>{{this}}</li>
{{/each}}
 </ul>
```

```
<ul class="people_list">
  <li>Yehuda Katz</li>
  <li>Alan Johnson</li>
  <li>Charles Jolley</li>
</ul>
```

BUILT-IN HELPERS

- **With**
  - It shifts the context for a section of a template

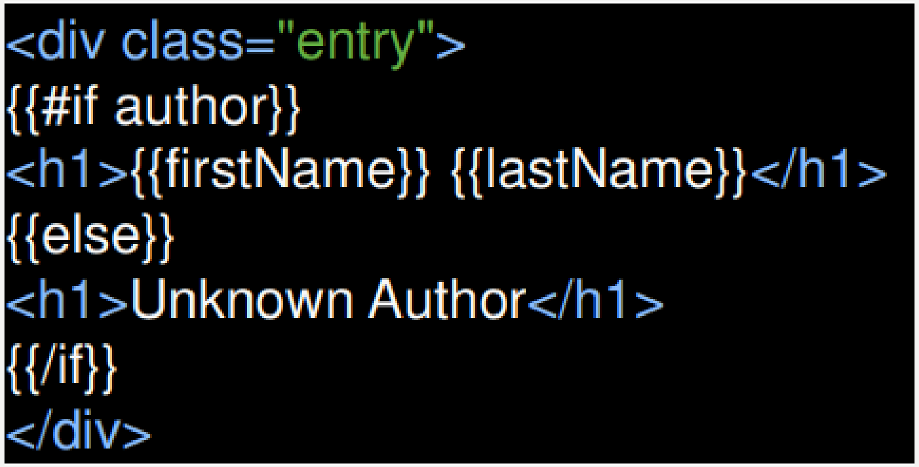
```
{ title: "My first post!",
  author: { firstName: "Charles", lastName: "Jolley" }
}
```

```
<div class="entry“>
<h1>{{title}}</h1>
{{#with author}}
<h2>By {{firstName}} {{lastName}}</h2>
{{/with}}
</div>
```

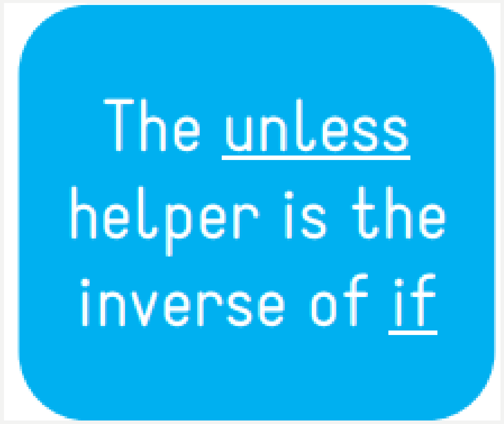
```
<div class="entry“>
<h1>My first post!</h1>
<h2>By Charles Jolley</h2>
</div>
```

BUILT-IN HELPERS

- **If - Else**
  - To conditionally render a block
  - It will render the block if its argument is not equal to false, undefined, null, []



```
<div class="entry">
{{#if author}}
<h1>{{firstName}} {{lastName}}</h1>
{{else}}
<h1>Unknown Author</h1>
{{/if}}
</div>
```



The unless helper is the inverse of if