handling user input

(web applications)
1. Display info (HTML / css)
2. Collect input (HTML forms)
3. Make use of input on client (javascript, callbacks)
   A. responsiveness (modify HTML with js, DOM)
   B. validation (modify input with js)
4. Send input to server (HTML forms + POST)
5. Store, package, and manipulate data (objects)
1. Display info (HTML / css)
2. Collect input (HTML forms)
3. Make use of input on client (javascript, callbacks)
   A. responsiveness (modify HTML with js, DOM)
   B. validation (modify input with js)
4. Send input to server (HTML forms + POST)
5. Store, package, and manipulate data (objects)
Baboons

Baboons are African and Arabian Old World monkeys belonging to the genus Papio, part of the subfamily Cercopithecinae. The five species are some of the largest non-hominoid members of the primate order; only the mandrill and the drill are larger.

Previously, the closely related gelada (genus Theropithecus) and the two species (mandrill and drill) of genus Mandrillus were grouped in the same genus, and these Old World monkeys are still often referred to as baboons in everyday speech. They range in size and weight depending on species. The Guinea baboon is 50 cm (20 in) and weighs only 14 kg (31 lb), while the largest
1. Display info (HTML / css)
2. Collect input (HTML forms)
3. Make use of input on client (javascript, callbacks)
   A. responsiveness (modify HTML with js, DOM)
   B. validation (modify input with js)
4. Send input to server (HTML forms + POST)
5. Store, package, and manipulate data (objects)
collecting input: forms
Forms (checkbox input)

```html
<form>
  <input type="checkbox" />
</form>
```
Forms (checkbox input)

<form>
<input type="checkbox"> (a checkbox)
</form>
Forms (checkbox input)

```html
<form>
  <input type="checkbox"/>
  (a checkbox)
</form>

(a checkbox)
```
Forms (checkbox input)

```html
<form>
  <input type="checkbox" />
</form>

(a checkbox)
```
Forms (checkbox input)

```
1 <form>
2     <input type="checkbox" checked> (a checkbox)
3 </form>
```

(a checkbox)
Forms (text input)

```
<form>
  <input type="text" size="30" value="default text">
</form>
```
Forms (text input)

```html
<form>
  <input type="text" size="30" value="default text">
</form>
```
Forms (password input)

```
<form>
  <input type="password" size="30">
</form>
```
Forms (password input)

```html
<form>
  <input type="password" size="30">
</form>
```
<form>
  <input type="radio" value="A" id="choice">
  <input type="radio" value="B" id="choice" checked>
  <input type="radio" value="C" id="choice">
  <input type="radio" value="D" id="choice">
</form>
<form>
  <input type="radio" value="A" id="choice"/>
  <input type="radio" value="B" id="choice" checked/>
  <input type="radio" value="C" id="choice"/>
  <input type="radio" value="D" id="choice"/>
</form>
Forms (radio input)

```html
<form>
  <input type="radio" value="A" id="choice"/>
  <input type="radio" value="B" id="choice" checked/>
  <input type="radio" value="C" id="choice"/>
  <input type="radio" value="D" id="choice"/>
</form>
```
Forms (radio input)

```html
<form>
  <input type="radio" value="A" id="choice">
  <input type="radio" value="B" id="choice" checked>
  <input type="radio" value="C" id="choice">
  <input type="radio" value="D" id="choice">
</form>
```
Forms (radio input)

```html
<form>
  <input type="radio" value="A" id="choice">
  <input type="radio" value="B" id="choice" checked>
  <input type="radio" value="C" id="choice">
  <input type="radio" value="D" id="choice">
</form>
```
Forms (file upload)

```html
<form>
  <input type="file">
</form>

Choose File  no file selected
```
<form>
  <input type="file" accept="image/*">
</form>
Forms (file upload)

```html
<form>
  <input type="file" accept="image/*" />
</form>
```

Files:
- test.html
- test.jpg
<form>
  <select>
    <option>artichoke</option>
    <option>eggplant</option>
    <option>mushrooms</option>
    <option>peppers</option>
  </select>
</form>
Forms (file upload)

```html
<button onclick="console.log('OK')">OK</button>
```

```
OK
```
Forms (file upload)

```html
<button onclick="console.log('OK')">OK</button>
```

callback
Forms (file upload)

```html
<button onclick="console.log('OK')">OK</button>
```
1. Display info (HTML / css)
2. Collect input (HTML forms)
3. Make use of input on client (javascript, callbacks)
   A. responsiveness (modify HTML with js, DOM)
   B. validation (modify input with js)
4. Send input to server (HTML forms + POST)
5. Store, package, and manipulate data (objects)
This is a simple web page that contains a script.

// some Javascript code
alert("There be dragons here!");
<html>
    <body>
        <p>This is a simple web page that contains a script.</p>
        <script>
            // some Javascript code
            alert("There be dragons here!");
        </script>
    </body>
</html>
Here are two pictures of the same baboon:

```html
<p>Here are two pictures of the same baboon:</p>
<img id="first_image" src="baboon.jpg" width="100">
<img id="second_image" src="baboon.jpg" width="100">

<script>
  var imgElement = document.getElementById("second_image");
  // make the second image small
  imgElement.width = "50";
</script>
```

Here are two pictures of the same baboon:
review: document object model (DOM)

browser gives js access to the HTML doc using `getElement`
review: responsiveness with callbacks
review: responsiveness with callbacks

```html
<html>
<body>
  <button onclick="toggleImage();">click</button>
  <br>
  <img id="im" src="baboon.jpg" height="300">
  
  <script>
  function toggleImage() {
    var visible = document.getElementById("im").style.display;
    if( visible === "none" ) {
      document.getElementById("im").style.display = "";
    } else {
      document.getElementById("im").style.display = "none";
    }
  }
  </script>
</body>
</html>
```
1. Display info (HTML / css)
2. Collect input (HTML forms)
3. Make use of input on client (javascript, callbacks)
   A. responsiveness (modify HTML with js, DOM)
   B. validation (modify input with js)
4. Send input to server (HTML forms + POST)
5. Store, package, and manipulate data (objects)
validation

1. client-side
   • part of responsive design
   • limit denial-of-service (DOS) attacks
   • libraries: Parsley.js, validate.js, Verify.js, gvalidator

2. server-side: limit data corruption attacks
1. Display info (HTML / css)
2. Collect input (HTML forms)
3. Make use of input on client (javascript, callbacks)
   A. responsiveness (modify HTML with js, DOM)
   B. validation (modify input with js)
4. Send input to server (HTML forms + POST)
5. Store, package, and manipulate data (objects)
Submitting forms (client-side)

```html
<form>
  Name: <input type="text" name="name"> <br>
  Password: <input type="password" name="password"> <br>
</form>
```
Submitting forms (client-side)

```html
<form>
  Name: <input type="text" name="name"/>
  <br/>
  Password: <input type="password" name="password"/>
  <br/>
</form>
```

Name: 
Password: 

name (server) vs. id (client)
Submitting forms (client-side)

```html
<form action="submit.php" method="POST">
  Name: <input type="text" name="name">  
  Password: <input type="password" name="password">  
  <button type="submit">Log in</button>
</form>
```
Submitting forms (client-side)

```html
<form action="submit.php" method="POST">
  Name: <input type="text" name="name"> <br>
  Password: <input type="password" name="password"> <br>
  <button type="submit">Log in</button>
</form>
```
Submitting forms (client-side)

```html
<form action="submit.php" method="POST">
  Name: <input type="text" name="name">  
  Password: <input type="password" name="password">
  <button type="submit">Log in</button>
</form>
```
Submitting forms (client-side)

```html
<form action="submit.php" method="POST">
  Name: <input type="text" name="name">
  <br>
  Password: <input type="password" name="password">
  <br>
  <button type="submit">Log in</button>
</form>
```
Submitting forms (client-side)

```html
<form action="submit.php" method="POST">
  Name: <input type="text" name="name"><br>
  Password: <input type="password" name="password"><br>
  <button type="submit">Log in</button>
</form>
```
Submitting forms (client-side)

```html
<form action="submit.php" method="POST">
  Name: <input type="text" name="name"/>
  Password: <input type="password" name="password"/>
  <button type="submit">Log in</button>
</form>
```
Submitting forms (client-side)

```html
<form action="submit.php" method="POST">
  Name: <input type="text" name="name">  
  Password: <input type="password" name="password">  
  <button type="submit">Log in</button>
</form>
```
Submitting forms (client-side)

```html
<form action="submit.php" method="POST">
  Name: <input type="text" name="name"/>
  Password: <input type="password" name="password"/>
  <button type="submit">Log in</button>
</form>

Welcome hanyfarid, your password has 12 characters
1. Display info (HTML / css)
2. Collect input (HTML forms)
3. Make use of input on client (javascript, callbacks)
   A. responsiveness (modify HTML with js, DOM)
   B. validation (modify input with js)
4. Send input to server (HTML forms + POST)
5. Store, package, and manipulate data (objects)
objects and OOP

1. objects: organize heterogeneous data (compound types)
2. OOP: organize code
   • organize data in objects
   • write code that deals with objects of a particular type
   • each object type gets its own js file
// create new empty objects for theDonlad and hillary
var theDonald = { }
var hillary = { }

theDonald.party = "Republican";
theDonald.hairColor = "orange";
theDonald.age = 69;

hillary.party = "Democrat";
hillary.hairColor = "blonde";
hillary.age = 68;

print( JSON.stringify(theDonald) )
{"party":"Republican","hairColor":"orange","age":69}
methods and OOP

(Pong example)