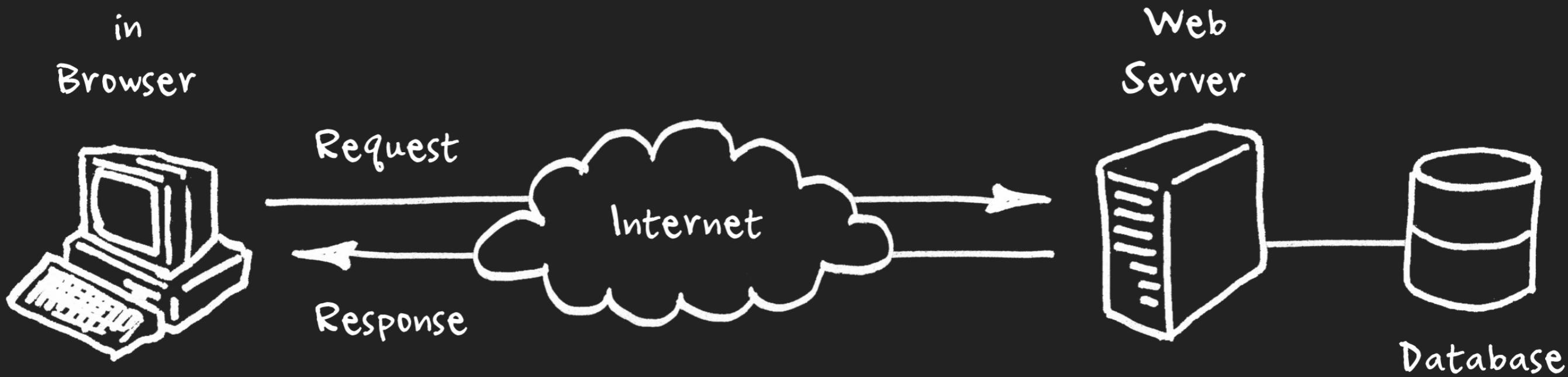


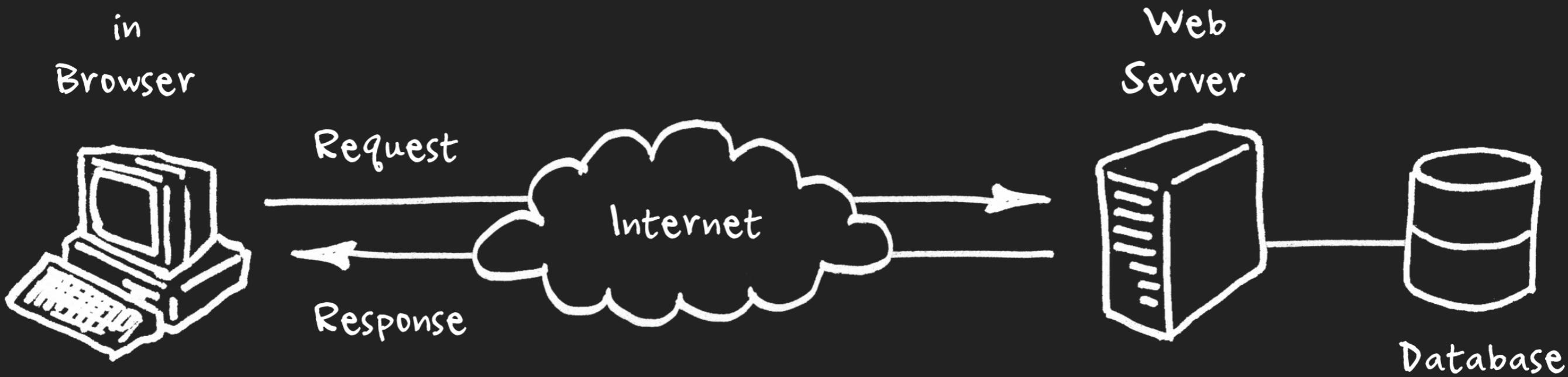
web applications

Web app  
in  
Browser



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)

Web app  
in  
Browser



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)

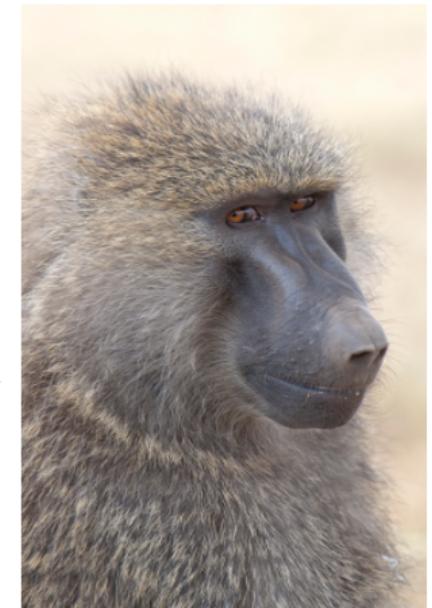
# review: displaying info

```
1 <!DOCTYPE html>
2 <html>
3   <head>
4     <title>Fundamentals of Web Programming</title>
5     <style type="text/css">
6       img {
7         float: right;
8       }
9     </style>
10    </head>
11    <body>
12
13    <h1>Baboons</h1>
14
15    
16
17    <p> 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
```

## 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



HTML: content  
CSS: style

## review: combining HTML and js

```
<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>
```

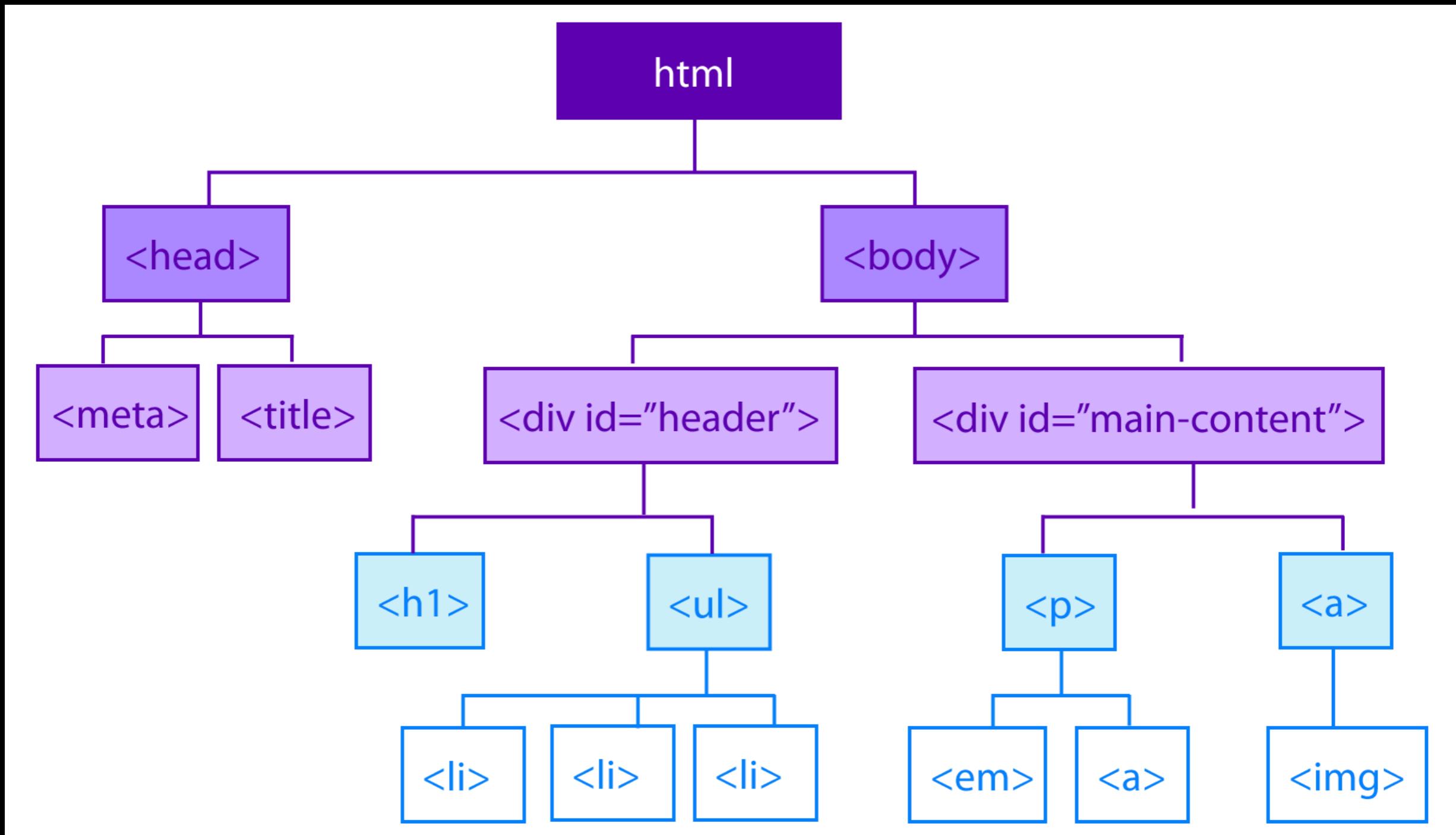
# review: combining HTML and js

```
<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>
```

# document object model (DOM)



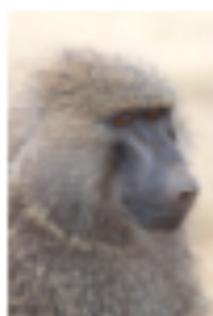
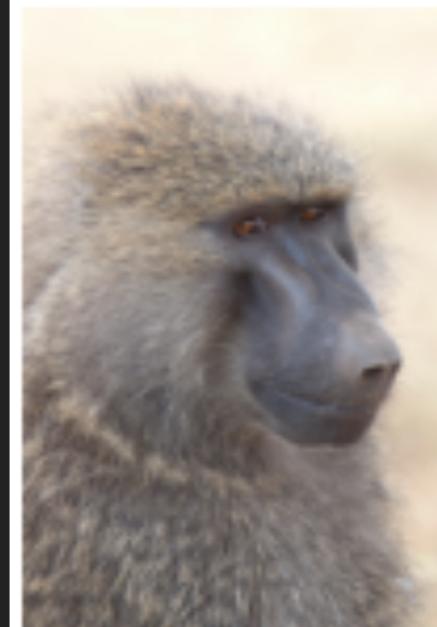
browser gives js access to the HTML doc using `getElement`

# review: modifying HTML with js

```
<html>
  <body>
    <p> Here are two pictures of the same baboon:</p>
    
    

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

Here are two pictures of the same baboon:



# Callbacks

```
<html>

<script>
  var clicked = function() {
    alert("Thank you for clicking");
  };
</script>

<body>
  <button onclick="clicked()">Click me!</button>
</body>

</html>
```

# Callbacks

```
<html>
```

```
  <script>
    var clicked = function() {
      alert("Thank you for clicking");
    };
  </script>
```

```
  <body>
```

```
    <button onclick="clicked();">Click me!</button>
  </body>
```

```
</html>
```

# Callbacks

```
<html>
```

```
  <script>
    var clicked = function() {
      alert("Thank you for clicking");
    };
  </script>
```

```
  <body>
    <button onclick="clicked()">Click me!</button>
  </body>
```

```
</html>
```

# Callbacks

```
<html>

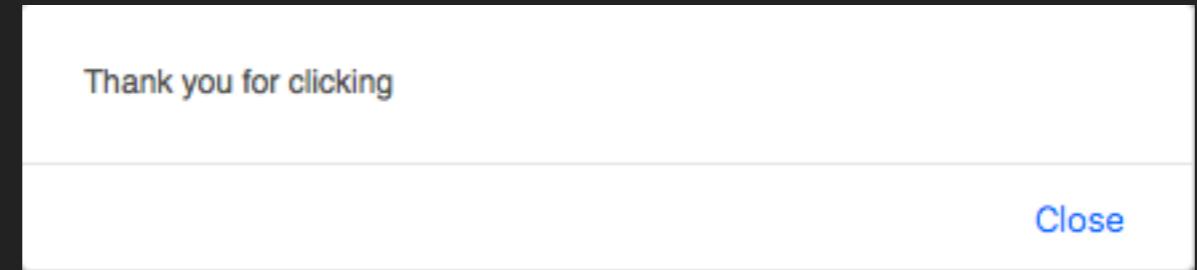
<script>
var clicked = function() {
    alert("Thank you for clicking");
};

</script>

<body>
    <button onclick="clicked()">Click me!</button>
</body>

</html>
```

# Callbacks



```
<html>
```

```
  <script>
    var clicked = function() {
      alert("Thank you for clicking");
    };
  </script>
```

```
  <body>
    <button onclick="clicked()">Click me!</button>
  </body>
```

```
</html>
```

# Callbacks

```
<html>
```

```
  <script>
    var clicked = function() {
      alert("Thank you for clicking");
    };
  </script>
```

```
  <body>
```

```
    <button onclick="clicked();">Click me!</button>
  </body>
```

```
</html>
```

# Callbacks

```
<html>
```

```
  <script>
    var clicked = function() {
      alert("Thank you for clicking");
    };
  </script>
```

```
  <body>
```

```
    <button onclick="clicked(); ">Click me!</button>
  </body>
```

```
</html>
```

## Exercise

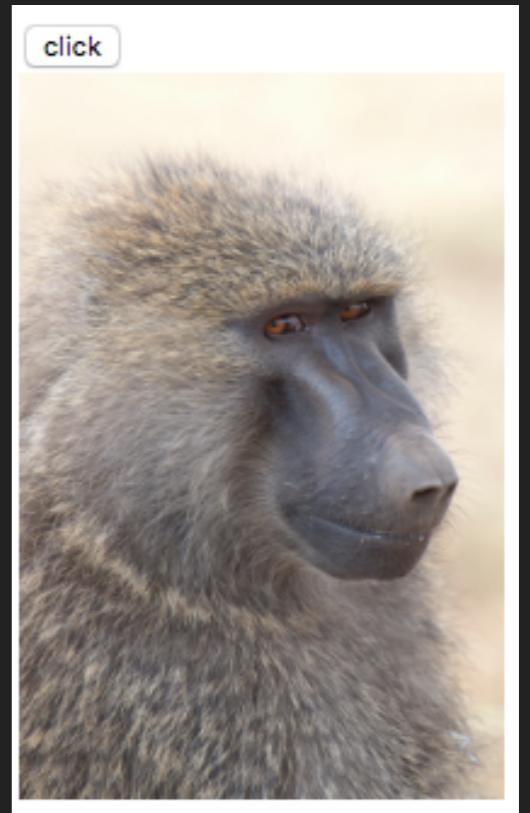
Create a webpage with a button and the baboon image. If the image is shown, then clicking the button will hide the image. If the image is not shown, then clicking the button will reveal the image.



Start slow and build up to your solution:

- 1) create a page with image and button.
- 2) write javascript function to show or hide image
- 3) ensure that clicking button calls the function

```
<html>
  <body>
    <button>click</button>
    <br>
    
  </body>
</html>
```



```
<html>
  <body>
    <button onclick="toggleImage()">click</button>
    <br>
    

    <script>
      function toggleImage() {
        document.getElementById("im").style.display = "none";
      }
    </script>
  </body>
</html>
```

```
<html>
<body>
    <button onclick="toggleImage()">click</button>
    <br>
    

    <script>
        var show = true;
        function toggleImage() {
            if( show ) {
                document.getElementById("im").style.display = "none";
                show = false;
            } else {
                document.getElementById("im").style.display = "";
                show = true;
            }
        }
    </script>
</body>
</html>
```

```
<html>
  <body>
    <button onclick="toggleImage()">click</button>
    <br>
    

    <script>
      var show = true;
      function toggleImage() {
        if( show ) {
          document.getElementById("im").style.display = "none";
        } else {
          document.getElementById("im").style.display = "";
        }
      }
    </script>
  </body>
</html>
```

```
<html>
  <body>
    <button onclick="toggleImage()">click</button>
    <br>
    

    <script>
      var show = true;
      function toggleImage() {
        if( show ) {
          document.getElementById("im").style.display = "none";
        } else {
          document.getElementById("im").style.display = "";
        }
        show = !show;
      }
    </script>
  </body>
</html>
```

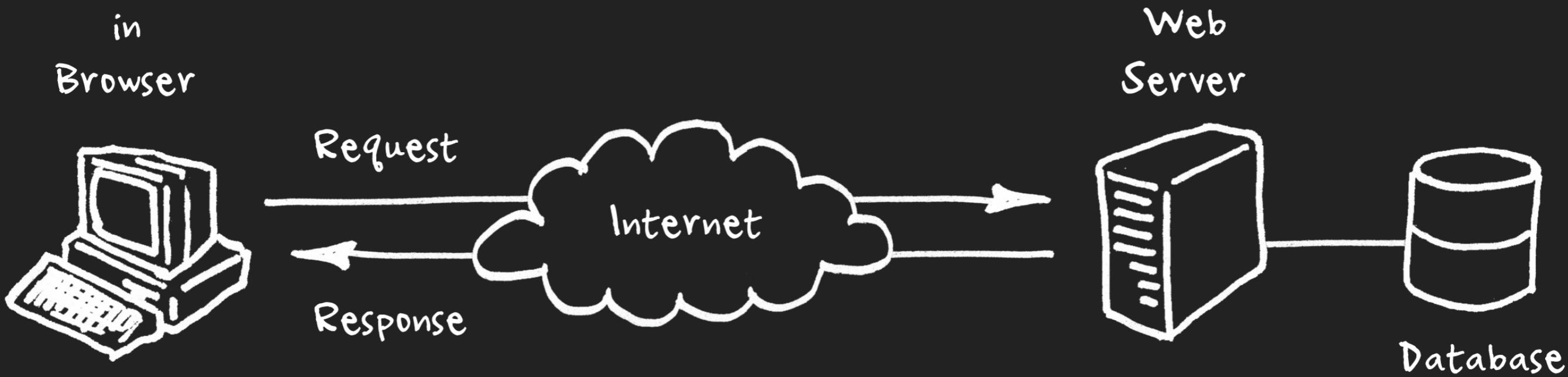
```
<html>
<body>
  <button onclick="toggleImage()">click</button>
  <br>
  

  <script>
    function toggleImage() {
      if (document.getElementById("im").style.display === "block")
        document.getElementById("im").style.display = "";
      else
        document.getElementById("im").style.display = "none";
    }
  </script>
</body>
</html>
```

```
<html>
<body>
  <button onclick="toggleImage()">click</button>
  <br>
  

  <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>
```

Web app  
in  
Browser



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

## A meeting room reservation form

Your name: \*

Your email address: \*

Department: \*

- select -

Which room would you like to reserve: \*

Room A  
 Room B  
 Room C

Extra requirements:

Flipchart and pens  
 Plasma TV screen  
 Coffee, tea and mineral water

Reservation date \*

Date format is YYYY-MM-DD

Reservation time :

09  00

## Forms (checkbox input)



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

## Forms (checkbox input)



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

## Forms (checkbox input)



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

## Forms (checkbox input)



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

(a checkbox)

## Forms (checkbox input)



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

(a checkbox)

## Forms (checkbox input)



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

(a checkbox)

## Forms (text input)



```
i 1 <form>  
2   <input type="text" size="30" value="default text">  
3 </form>
```

default text

## Forms (text input)



```
i 1 <form>  
2   <input type="text" size="30" value="default text">  
3 </form>
```

default text

## Forms (password input)



```
i 1 <form>
2     <input type="password" size="30">
3 </form>
```

## Forms (password input)



```
i 1 <form>  
2     <input type="password" size="30">  
3 </form>
```

A screenshot of a web browser showing a password input field. The field contains five black dots ('.....') followed by a cursor. To the right of the input field is a small icon of a key with a dropdown arrow, typically used for password strength indicators or copy/clear functions.

## Forms (radio input)



```
i 1 <form>
 2   <input type="radio" value="A" id="choice">
 3   <input type="radio" value="B" id="choice" checked>
 4   <input type="radio" value="C" id="choice">
 5   <input type="radio" value="D" id="choice">
 6 </form>
```

## Forms (radio input)



```
i 1 <form>  
2     <input type="radio" value="A" id="choice">  
3     <input type="radio" value="B" id="choice" checked>  
4     <input type="radio" value="C" id="choice">  
5     <input type="radio" value="D" id="choice">  
6 </form>
```

## Forms (radio input)



```
i 1 <form>  
2     <input type="radio" value="A" id="choice">  
3     <input type="radio" value="B" id="choice" checked>  
4     <input type="radio" value="C" id="choice">  
5     <input type="radio" value="D" id="choice">  
6 </form>
```

## Forms (radio input)



```
i 1 <form>  
2   <input type="radio" value="A" id="choice">  
3   <input type="radio" value="B" id="choice" checked>  
4   <input type="radio" value="C" id="choice">  
5   <input type="radio" value="D" id="choice">  
6 </form>
```

A horizontal row of four radio buttons. The second button from the left is filled with blue and has a white dot in the center, indicating it is selected. The other three buttons are empty circles.

## Forms (radio input)



```
i 1 <form>
2   <input type="radio" value="A" id="choice">
3   <input type="radio" value="B" id="choice" checked>
4   <input type="radio" value="C" id="choice">
5   <input type="radio" value="D" id="choice">
6 </form>
```

A horizontal row of five radio buttons. The first four are white circles with gray outlines, while the fifth one is a blue circle with a white dot, indicating it is selected.

## Forms (file upload)



```
i 1 <form>
2     <input type="file">
3 </form>
```

Choose File no file selected

## Forms (file upload)

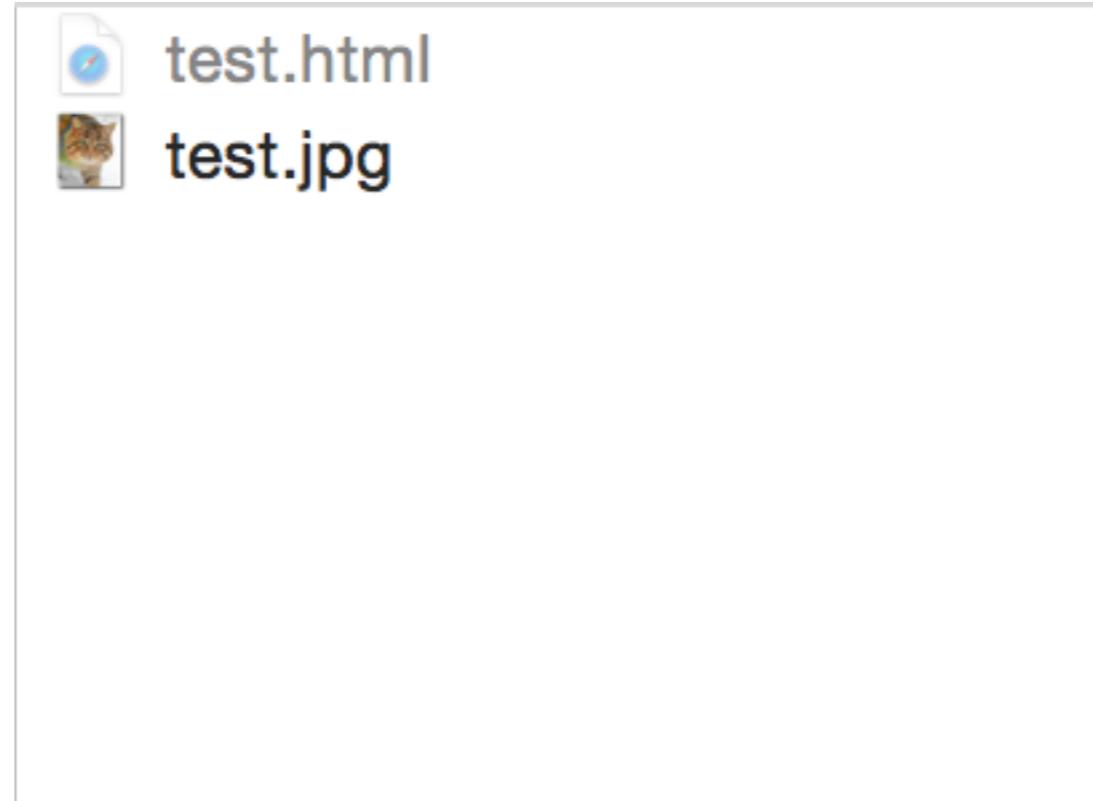


```
i 1 <form>
 2   <input type="file" accept="image/*">
 3 </form>
```

## Forms (file upload)



```
i 1 <form>  
2   <input type="file" accept="image/*">  
3 </form>
```



## Forms (select input)



```
i 1 <form>
2   <select>
3     <option>artichoke</option>
4     <option>eggplant</option>
5     <option>mushrooms</option>
6     <option>peppers</option>
7   </select>
8 </form>
```

A screenshot of a user interface showing a dropdown menu. The menu is a white box with a thin black border. It contains four items: "artichoke", "eggplant", "mushrooms", and "peppers". The item "artichoke" is highlighted with a blue background and has a white checkmark icon to its left. The other three items are in a standard black font.

## Forms (buttons)



```
i 1 <button onclick="console.log('OK')>OK</button>
```

OK

## Forms (file upload)



i 1 <button onclick="console.log('OK')>OK</button>

OK

callback

# Forms (file upload)

A screenshot of a browser developer tools console. On the left, there is a code editor window showing a single line of code: <button onclick="console.log('OK')>OK</button>. To the right of the code editor is a modal dialog box with a single button labeled "OK". A large black arrow points to the "OK" button, indicating it is being clicked. Below the code editor and modal are the standard browser developer tools tabs: Elements, Network, Resources, Timelines, Debug, Storage, Console, and Search. The Console tab is selected. At the bottom of the console, there is a status bar with the text "OK" on the left and "onclick — lecture07.html:1" on the right, along with a "Main Frame" dropdown.

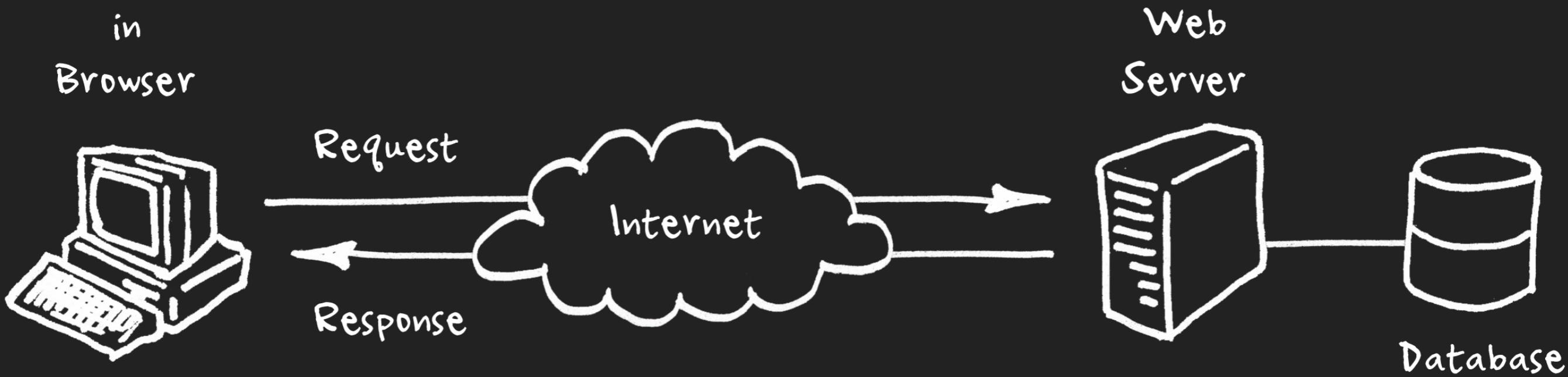
```
<button onclick="console.log('OK')>OK</button>
```

OK

Elements Network Resources Timelines Debug Storage Console Search

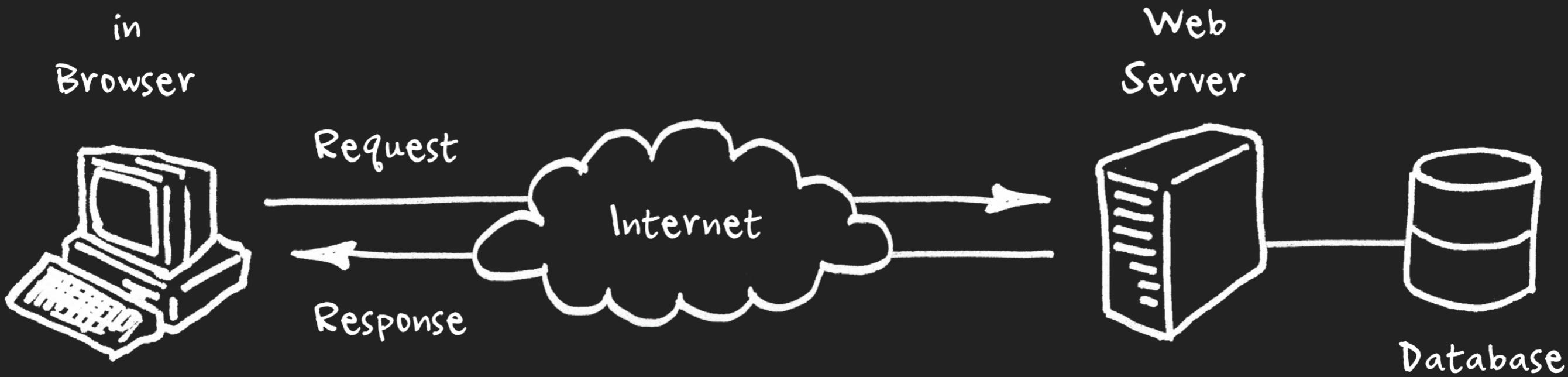
OK onclick — lecture07.html:1 Main Frame

Web app  
in  
Browser



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)

Web app  
in  
Browser

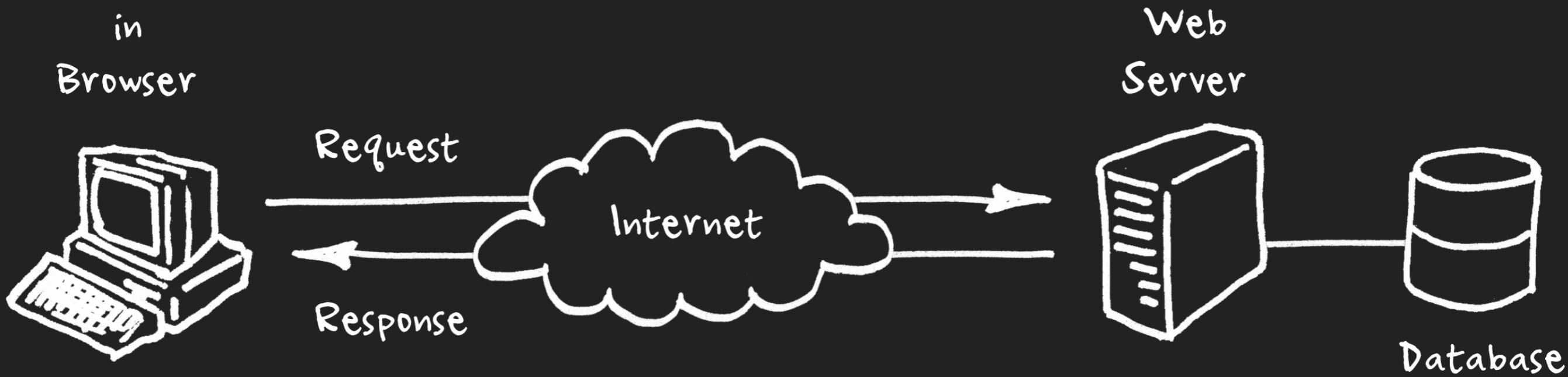


1. Display info (HTML / css)
2. Collect input (HTML forms)
3. Make use of input on client (javascript, callbacks)
  - A. responsiveness (modify HTML with js, DOM)
  - B. validation (modify input with js)
4. Send input to server (HTML forms + POST)
5. Store, package, and manipulate data (objects)

# 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

Web app  
in  
Browser



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)



```
i 1 <form>
2   Name: <input type="text" name="name"> <br>
3   Password: <input type="password" name="password"> <br>
4 </form>
```

Name:

Password:

## Submitting forms (client-side)



```
i 1 <form>
2   Name: <input type="text" name="name"> <br>
3   Password: <input type="password" name="password"> <br>
4 </form>
```

Name:

Password:

name (server) vs. id (client)

## Submitting forms (client-side)



```
i 1 <form action="submit.php" method="POST">
 2   Name: <input type="text" name="name"> <br>
 3   Password: <input type="password" name="password"> <br>
 4   <button type="submit">Log in</button>
 5 </form>
```

## Submitting forms (client-side)



```
i 1 <form action="submit.php" method="POST">
2   Name: <input type="text" name="name"> <br>
3   Password: <input type="password" name="password"> <br>
4   <button type="submit">Log in</button>
5 </form>
```

## Submitting forms (client-side)



```
i 1 <form action="submit.php" method="POST">
2   Name: <input type="text" name="name"> <br>
3   Password: <input type="password" name="password"> <br>
4   <button type="submit">Log in</button>
5 </form>
```

## Submitting forms (client-side)



```
i 1 <form action="submit.php" method="POST">
2   Name: <input type="text" name="name"> <br>
3   Password: <input type="password" name="password"> <br>
4   <button type="submit">Log in</button>
5 </form>
```

## Submitting forms (client-side)



```
i 1 <form action="submit.php" method="POST">
2   Name: <input type="text" name="name"> <br>
3   Password: <input type="password" name="password"> <br>
4   <button type="submit">Log in</button>
5 </form>
```

Name:

Password:

Log in

## Submitting forms (client-side)



```
i 1 <form action="submit.php" method="POST">
2   Name: <input type="text" name="name"> <br>
3   Password: <input type="password" name="password"> <br>
4   <button type="submit">Log in</button>
5 </form>
```

Name:

Password:

**Log in**

## Submitting forms (client-side)



```
i 1 <form action="submit.php" method="POST">
2   Name: <input type="text" name="name"> <br>
3   Password: <input type="password" name="password"> <br>
4   <button type="submit">Log in</button>
5 </form>
```

Name: hanyfarid

Password: .....

**Log in**

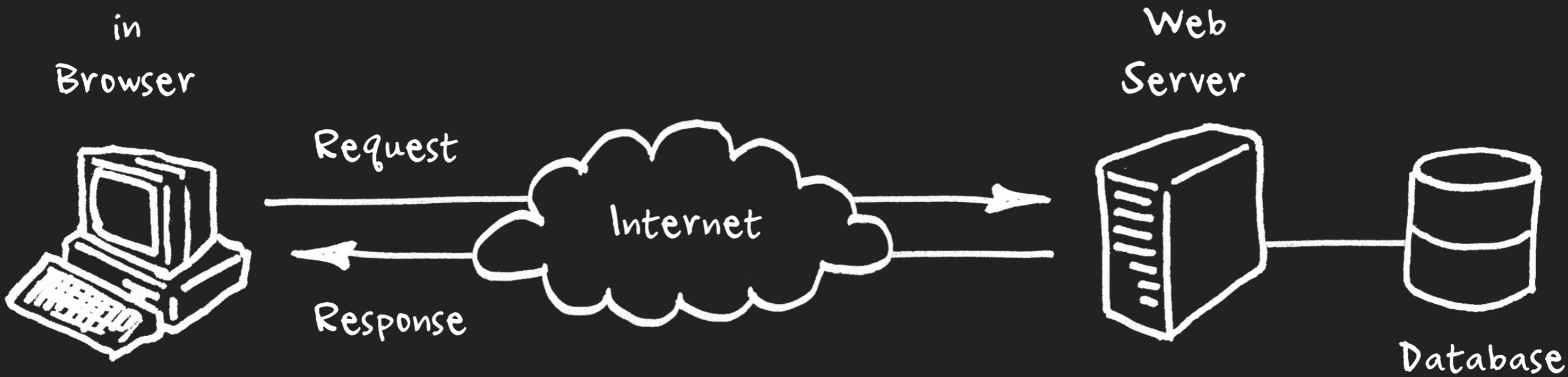
## Submitting forms (client-side)



```
i 1 <form action="submit.php" method="POST">
 2   Name: <input type="text" name="name"> <br>
 3   Password: <input type="password" name="password"> <br>
 4   <button type="submit">Log in</button>
 5 </form>
```

Welcome hanyfarid, your password has 12 characters

Web app  
in  
Browser



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 and manipulate data server-side (PHP, etc)