Part 1: Events continued

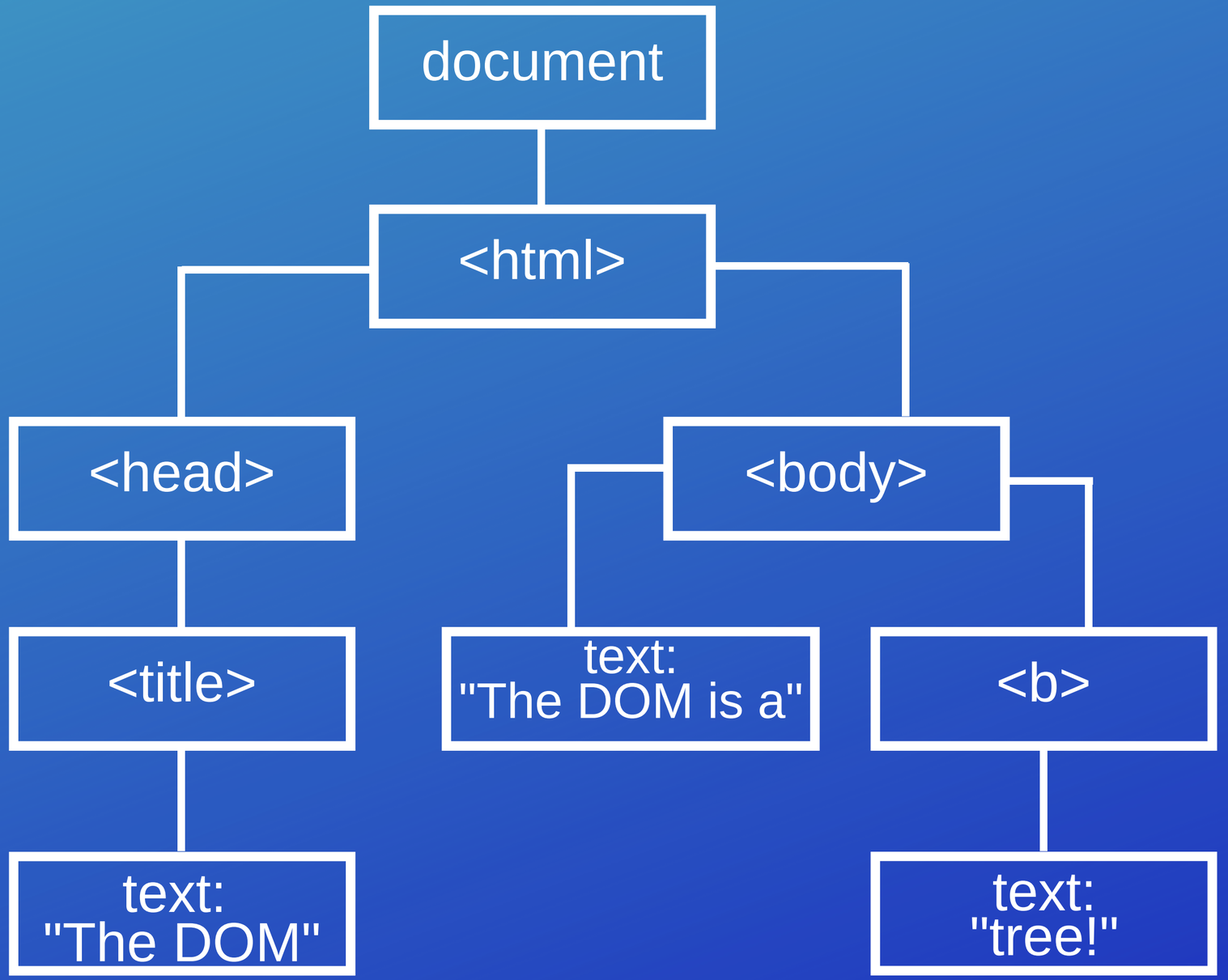
Anna Azzam | Frontend Developer | anna.a@canva.com

Recap

1) The DOM

2) Events

3) Event Handlers



Recap

1) The DOM

2) Events

3) Event Handlers

An *event* is a signal that a "thing" has happened to a DOM element

We can use events to add user interaction to our application

Recap

1) The DOM

2) Events

3) Event Handlers

```
let element = document.getElementById('btn');
let handler = (event) => {
  alert('button was clicked');
})

element.addEventListener('click', handler);
element.removeEventListener('click', handler);
```

Basketball drag and drop game

CSS Animations

CSS has a wide variety of animations available. There are a number of properties you can use to control what your animation looks like:

```
animation-name // custom
animation-duration // length of the animation e.g. 2s
animation-delay // delay before the animation e.g. 2s
animation-iteration-count // e.g. 2, infinite
animation-direction // e.g. normal, alternate, reverse
animation-timing-function // e.g. ease, linear
animation-fill-mode // e.g. forwards, backwards
animation // you can use the shorthand
```

Keyframes

Keyframes are a way of specifying the values of CSS properties at intermediate steps throughout an animation.

```
@keyframes slidein {
  from {
    margin-left: 100%;
    width: 300%;
  }
  to {
    margin-left: 0%;
    width: 100%;
  }
}
```

```
@keyframes identifier {
  0% { top: 0; }
  50% { top: 30px; }
  50% { top: 10px; }
  100% { top: 0; }
}
```

e.g. animate opacity for a fade effect

```
@keyframes fadein {
  from {
    opacity: 0;
  }
  to {
    opacity: 1;
  }
}

animation: fadein 0.4s ease-in-out
```

e.g. animate scale for a zoom effect

```
@keyframes zoom {
  from {
    transform: scale(1.5);
  }
  to {
    transform: scale(1);
  }
}

animation: fade .3s ease-in reverse;
```

e.g. animate rotation for a spin effect

```
@keyframes rotate {
  from {
    transform: rotate(360deg);
  }
  to {
    transform: rotate(0deg);
  }
}

animation: rotation 0.7s ease-in;
```

CSS Animations Example

Adding animations to our basketball game

Part 2: Forms

Forms

- A HTML <form> element is a way of defining a form which is used to get user input
- They consist of different types of input elements:
  - text fields
  - checkboxes
  - radio buttons
  - submit buttons
- We specify the type of input element using the type attribute: `<input type="text">`

```
<form>
  First name:<br>
  <input type="text" name="firstname">
  Last name:<br>
  <input type="text" name="lastname">
</form>
```

document.forms

When you have forms in your document, they can be found in a special document property called `document.forms`

This is a “named collection”, i.e. it’s both named and ordered. We can use both the name or the number in the document to get the form.

```
document.forms.test // the form with name="test"
document.forms["test"] // also the form with name="test"
document.forms[0] // the first form in the document
```

form.elements

Each form has a field `form.elements` which has all of the elements in that form. This is also a “named collection”

HTML

```html
<form>
  <input type="text" name="fname">
  <input type="radio" name="age" value="10">
  <input type="radio" name="age" value="20">
</form>
```

JS

```js
const form = document.forms[0];

// element with the name "fname"
form.elements.fname;
// shorter notation:
form.fname;

// since there are multiple elements with the name "age", this returns a collection
const ages = form.elements.age;
```

Backreferences

Each form element stores a backreference to the form it came from, `element.form`

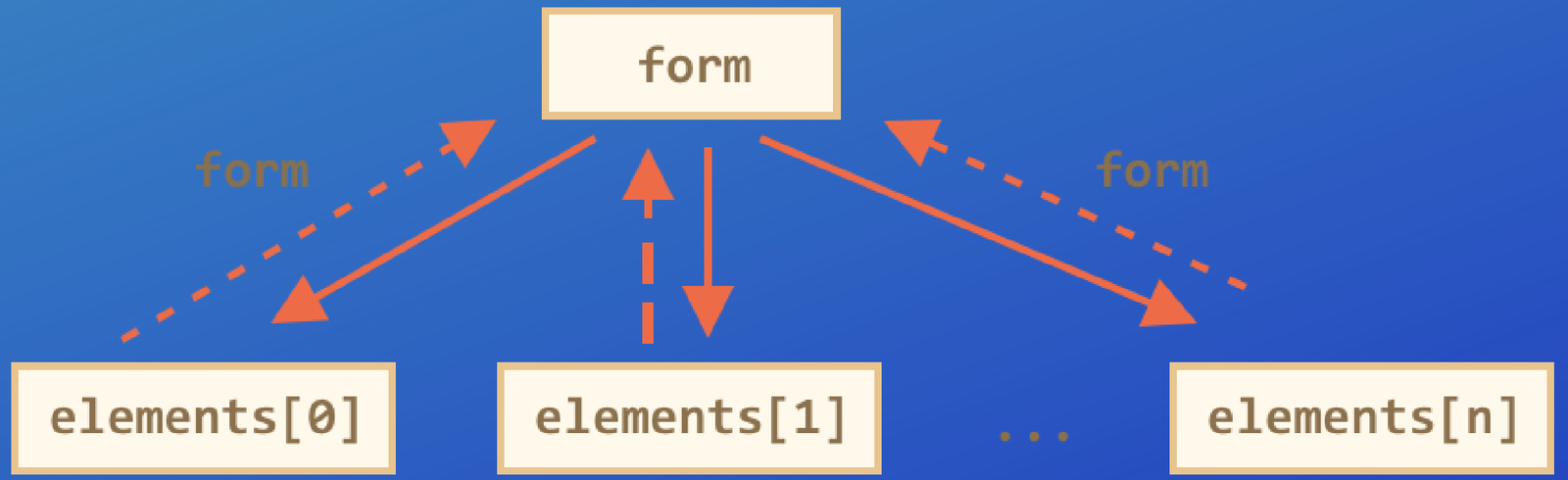


Image source: javascript.info

Backreferences

HTML

```html
<form>
  <input type="text" name="login">
</form>
```

JS

```js
const form = document.forms[0];
const login = form.login;

console.log(login.form)
```

What will this console.log?

Form Values

To get the value of a form element:

```
// To get the text for an input element or textarea:
input.value

// To get a boolean for a checkbox or radio button
input.checked

// For a <select> tag
select.options // the collection of <option>s
select.value // the value of the currently selected option,
select.selectedIndex // the index of the currently selected option
```

Submit Buttons and onsubmit

- <input type="submit"> defines a button for submitting the form data to a form-handler
- Clicking a submit button triggers a "submit" event on the form
- We can listen to this event, and run form validation

```
form.addEventListener('submit', (event) => {
  event.preventDefault();
  // form validation can go here
});
```

Login Form Example

Part 3: Localstorage

What is Local Storage?

- Window.localStorage is an API that allows you to read and write to a storage object in the document
- The stored data in this storage object is persisted between sessions
- This means we can save and retrieve data when a user closes their tab or browser

Localstorage API

```
// Add a data item given the key and value
localStorage.setItem(key, value);

// Retrieves an item from localstorage given a key
const value = localStorage.getItem(key);

// Remove an item with a given key from localstorage
localStorage.removeItem(key);

// Remove all items from localstorage
localStorage.clear();
```

Adding localStorage to our form

Thank you!

Anna Azzam | Frontend Developer | anna.a@canva.com