ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ
ATHENS UNIVERSITY OF ECONOMICS AND BUSINESS
ΚΕΝΤΡΟ ΕΠΙΜΟΡΦΩΣΗΣ ΚΑΙ ΔΙΑ ΒΙΟΥ ΜΑΘΗΣΗΣ

React JS
Components & Props

Δαμιανός Μαρινάτος

Components (1)

- Κάθε **Component** είναι μία **ανεξάρτητη μονάδα** που **ενθυλακώνει**:
  - τη **δομή** (HTML/JSX),
  - την **εμφάνιση** (CSS),
  - και τη **συμπεριφορά** (JavaScript logic).

  Με αυτόν τον τρόπο, ένα Component μπορεί να χρησιμοποιηθεί ξανά σε διαφορετικά σημεία της εφαρμογής και να συνδυαστεί με άλλα Components για τη δημιουργία πιο σύνθετων UI.

Components (2)

- Η React υποστηρίζει **δύο βασικούς τύπους components**:
  - **Class Components**
  - **Functional Components**

Class Components (1)

- Είναι οι "παραδοσιακές" κλάσεις της JavaScript.
- Υποστηρίζουν state και lifecycle methods μέσω της class σύνταξης όπως componentDidMount και componentWillUnmount.
- Χρησιμοποιούνται λιγότερο στις νέες εφαρμογές.

Class Components (2)


```jsx
import { Component } from "react";

class ClassComponent extends Component {  Show usages
  render() : Element  {  no usages
    const title = "I'm a class component!";
    return <h1>{title}</h1>;
  }
}

export default ClassComponent;  Show usages
```

```jsx
import ClassComponent from "../components/ClassComponent";

export default function Page() : Element  {  Show usages
  return (
    <>
      <ClassComponent />
    </>
  );
}
```

Class Components (3)


```tsx
import { Component } from "react";

type Props = object;
type State = {
  title: string;
};
class ClassAdvancedComponent extends Component<Props, State> {  Show usages
  constructor(props: Props) {  no usages
    super(props);
    this.state = {
      title: "I'm a class component!",
    };
  }
  componentDidMount() : void  {  no usages
    console.log( ...data: "Component mounted!");
  }
  componentWillUnmount() : void  {  no usages
    console.log( ...data: "Component will unmount!");
  }
  render() : Element  {  no usages
    return <h1>{this.state.title}</h1>;
  }
}

export default ClassAdvancedComponent;  Show usages
```

Functional Components (1)

- Είναι απλές συναρτήσεις που επιστρέφουν JSX.
- Με την εισαγωγή των **Hooks**, μπορούν πλέον και αυτές να έχουν state και side effects.
- Είναι ο πιο σύγχρονος και προτιμώμενος τρόπος ανάπτυξης components.

Functional Components (2)


```
function FunctionalComponent() : Element  {  Show usages
  const title = "I'm a functional component!";
  return <h1>{title}</h1>;
}
export default FunctionalComponent;  Show usages
```

```
import FunctionalComponent from "../components/FunctionalComponent";

export default function PublicPage() : Element  {  Show usages
  return (
    <>
      <FunctionalComponent />
    </>
  );
}
```

Arrow Function Components

- Είναι μια **παραλλαγή των Functional Components** που γράφονται με τη σύνταξη βέλους (=>).
- Παρέχουν **πιο σύντομη και καθαρή σύνταξη**, ειδικά για μικρά και απλά components.
- Συμπεριφέρονται **ακριβώς όπως τα κανονικά Functional Components**.

Arrow Function Components


```
const ArrowFunctionalComponent : () => Element = () : Element => {  Show usages
  const title = "I'm an arrow functional component!";
  return <h1>{title}</h1>;
};
export default ArrowFunctionalComponent;  Show usages
```

```
import ArrowFunctionalComponent from "../components/ArrowFunctionalComponent";

export default function PublicPage() : Element {  Show usages
  return (
    <>
      <ArrowFunctionalComponent />
    </>
  );
}
```

Arrow Function Components


```
const ArrowFunctionalComponent : () => Element = () : Element => {  Show usages
  const title = "I'm an arrow functional component!";
  return <h1>{title}</h1>;
};
export default ArrowFunctionalComponent;  Show usages
```

```
import ArrowFunctionalComponent from "../components/ArrowFunctionalComponent";

export default function PublicPage() : Element {  Show usages
  return (
    <>
      <ArrowFunctionalComponent />
    </>
  );
}
```

Props (1)

- Τα **props** που χρησιμοποιούνται για να περάσουμε πληροφορίες από ένα component σε ένα άλλο.
- Είναι **μόνο για ανάγνωση** (read-only): δεν επιτρέπεται να τροποποιούνται από το parent component.
- Χρησιμοποιούνται για να επαναχρησιμοποιούμε και να παραμετροποιούμε components με διαφορετικά δεδομένα.

Props (2)


```tsx
type Props = {
  title: string;
};

const ArrowFunctionalComponentWithProps : ({ title }: Props) => Element  = ({ title }: Props) : Element  => {
  return <h1>{title}</h1>;
};
export default ArrowFunctionalComponentWithProps;  Show usages
```

```tsx
import ArrowFunctionalComponentWithProps from "../components/ArrowFunctionalComponentWithProps";

export default function PublicPage() : Element  {  Show usages
  const title = "I'm an arrow functional component with props!";
  return (
    <>
      <ArrowFunctionalComponentWithProps title={title} />
    </>
  );
}
```

Type vs Interface

- Στην TypeScript, οι λέξεις-κλειδιά **type** και **interface** χρησιμοποιούνται για να ορίσουμε τύπους (types).

```
type Props = {
  title: string;
};
```

```
interface Props {
  title: string;
}
```

Type

- Μπορεί να χρησιμοποιηθεί για **τύπους πιο γενικούς** (π.χ. union, tuple, function types).
- Δεν υποστηρίζει declaration merging.

```
type A = { title: string };
type B = { description: string };
type Props = A & B;

const ArrowFunctionalComponentWithProps : ({ title, description }: Props) => Elemen...  = ({ title, description }: Props) : Element  => {
  return (
    <>
      <h1>{title}</h1>
      <p>{description}</p>
    </>
  );
};

export default ArrowFunctionalComponentWithProps; Show usages  new *
```

Interface

- Σχεδιάστηκε κυρίως για **αντικείμενα** και **πιο κλασική OOP χρήση** (Object-Oriented Programming).
- Υποστηρίζει **declaration merging**, δηλαδή μπορούμε να ορίσουμε ξανά το ίδιο interface και να προσθέσουμε πεδία.

```
interface Props {  Show usages  new *
  title: string;
}
interface Props {  Show usages  new *
  description: string;
}

const ArrowFunctionalComponentWithProps : ({ title, description }: Props) => Elemen...  = ({ title, description }: Props) : Element  => {
  return (
    <>
      <h1>{title}</h1>
      <p>{description}</p>
    </>
  );
};

export default ArrowFunctionalComponentWithProps;  Show usages  new *
```

Children Prop (1)

- Το children είναι ένα ειδικό prop που επιτρέπει σε ένα React component να περιέχει και να εμφανίζει δυναμικά οποιοδήποτε περιεχόμενο τοποθετηθεί ανάμεσά του κατά την κλήση.
- Ιδανικό για components που "τυλίγουν" άλλα στοιχεία, χωρίς να γνωρίζουν εκ των προτέρων το περιεχόμενο.

```tsx
import React from "react";
import Header from "./Header";
import Footer from "./Footer";

interface LayoutProps {  Show usages  new *
  children: React.ReactNode;
}

const Layout : ({ children }: LayoutProps) => Element  = ({ children }: LayoutProps) : Element  => {
  return (
    <>
      <Header />
      <div className="container mx-auto min-h-96 px-4 sm:px-6 lg:px-8">
        {children}
      </div>
      <Footer />
    </>
  );
};
```

Children Prop (2)

- Το children έχει τύπο ReactNode, για να μπορούμε να περάσουμε οποιοδήποτε έγκυρο JSX περιεχόμενο.

```tsx
import React from "react";
import Header from "./Header";
import Footer from "./Footer";

interface LayoutProps {  Show usages  new *
  children: React.ReactNode;
}

const Layout : ({ children }: LayoutProps) => Element  = ({ children }: LayoutProps) : Element  => {
  return (
    <>
      <Header />
      <div className="container mx-auto min-h-96 px-4 sm:px-6 lg:px-8">
        {children}
      </div>
      <Footer />
    </>
  );
};
```

Component Composition (1)

- Η React επιτρέπει τη σύνθεση πολλών μικρών components για να δημιουργήσουμε πιο σύνθετες διεπαφές.
- **Composition** σημαίνει ότι φτιάχνουμε components που περιλαμβάνουν άλλα components.
- Αυτό μας βοηθάει να **επαναχρησιμοποιούμε** κώδικα και να κρατάμε τη λογική **modular** και ευανάγνωστη.

Component Composition (2)


```tsx
type CardProps = {
  title: string;
  children: React.ReactNode;
};

const Card : ({ title, children }: CardProps) => Eleme...  = ({ title, children }: CardProps) : Element  => {
  return (
    <div className="card">
      <h2>{title}</h2>
      {children}
    </div>
  );
};

export default Card;  no usages  new *
```

```tsx
<Card title="Component CompositionTitle">
  <p>Αυτό είναι το περιεχόμενο της κάρτας.</p>
</Card>
```